
Metadata Registration API

Release 0.5.0

Rafael S. Müller

Feb 08, 2022

CONTENTS:

1	Installation	3
1.1	Prerequisites	3
1.2	Setup	3
2	Getting started	5
2.1	Resources	5
2.2	Access	5
3	Metadata_Registration_API	7
3.1	metadata_registration_api package	7
3.2	scripts package	23
3.3	test package	23
4	Indices and tables	25
	Python Module Index	27
	Index	29

A microservice which provides an API to access the Metadata Registration Tool.

INSTALLATION

This chapter describes how to install and setup the metadata registration API (hereafter API). Before you can install the project, make sure that you meet the requisites:

1.1 Prerequisites

Database The API uses MongoDB as data store. Independent on how you install the API, make sure you have the necessary credentials to access your MongoDB instance.

Dependencies The API has two internal dependencies. These dependencies are the ‘State Machine’ and the ‘Dynamic Form’. Both projects are listed as git repositories in the *requirements.txt* document. To download them, you must be able to access github with a SSH key. Check [here](#) for a better description.

Credentials / Configuration A configuration done through environment variables, make sure they’re all defined as needed.

1.2 Setup

The API can be installed in three different ways:

- *As a podman container*
- *As a docker container*
- *In a conda environment*

Although all three approaches work, we recommend running it either as a docker or podman container since it usually takes longer and is more error prone when installing the API in a conda environment.

1.2.1 Run as podman container

After, we have met the prerequisites, clone the git repository into our preferred directory. Change into the root directory of the project and copy the ssh private key (i. e. `id_rsa`) and the `.credentials.yaml`. Finally execute the following command:

```
$ ./start_podman.sh
```

The script checks if an identical container is already running. If this is the case, the container is stopped and removed, and a new container is build and executed.

1.2.2 Run as docker container

After, we have met the prerequisites, clone the git repository into our preferred directory. Change into the root directory of the project and copy the ssh private key (i. e. `id_rsa`) and the `.credentials.yaml`. Finally execute the following command:

```
$ ./start_docker.sh
```

The script checks if an identical container is already running. If this is the case, the container is stopped and removed, and a new container is build and executed.

1.2.3 Run in a conda environment

[Documentation not available]

GETTING STARTED

The API consists of multiple resources. An overview of the resources is available as swagger documentation. The swagger documentation is accessible via the index of the url. If you run the software locally, the swagger documentation is found under ``http://127.0.0.1:5001``.

2.1 Resources

The API has several resources including *properties*, *controlled vocabularies*, *forms*, *studies* and *users*. The API follows the REST principle.

2.2 Access

Some API commands require an access token. As a registered user you can generate an access token by sending your email address and password as a post request to ``user/login`_`. The server will response with an access token.

The access token has to be transmitted in the header of each restricted request as value of *x-access-token*.

```
{
  "x-access-token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyZXNlIjoiaWNRmMjMyZmZlIiwiaWF0IjoxNTc4NTg3Mjc4LjE1Nzg1ODkwNzN9.PGB-jyqdcpz3LmwOeMp1rxa9a7tVizHqh3EceFWy9dI"}

```

The access token is only limited for a limited time span. Afterwards, a new token has to be generated.

METADATA_REGISTRATION_API

3.1 metadata_registration_api package

3.1.1 Subpackages

metadata_registration_api.api package

Submodules

metadata_registration_api.api.api_ctrl_voc module

```
class metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary(api=None, *args,  
                                                                           **kwargs)
```

Bases: flask_restx.resource.Resource

delete(*user=None*)
Delete all entries

get()
Fetch a list with all entries

get_parser = <flask_restx.reqparse.RequestParser object>

methods: Optional[List[str]] = {'DELETE', 'GET', 'POST'}
A list of methods this view can handle.

post(*user=None*)
Add a new entry

```
class metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabularyId(api=None, *args,  
                                                                           **kwargs)
```

Bases: flask_restx.resource.Resource

delete(*id, user=None*)
Delete an entry given its unique identifier

get(*id*)
Fetch an entry given its unique identifier

methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}
A list of methods this view can handle.

put(*id, user=None*)
Update an entry given its unique identifier

```
class metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabularyItemsMap(api=None,
                                                                              *args,
                                                                              **kwargs)
```

Bases: flask_restx.resource.Resource

get()

Get a map for CV items: cv_name: {item_key: item_value}

methods: Optional[List[str]] = {'GET'}

A list of methods this view can handle.

metadata_registration_api.api.api_form module

```
class metadata_registration_api.api.api_form.ApiForm(api=None, *args, **kwargs)
```

Bases: flask_restx.resource.Resource

delete(*user=None*)

Delete all entries

get()

Fetch a list with all entries

get_parser = <flask_restx.reqparse.RequestParser object>

methods: Optional[List[str]] = {'DELETE', 'GET', 'POST'}

A list of methods this view can handle.

post(*user=None*)

Add a new entry

```
class metadata_registration_api.api.api_form.ApiFormId(api=None, *args, **kwargs)
```

Bases: flask_restx.resource.Resource

delete(*id, user=None*)

Delete an entry given its unique identifier

get(*id*)

Fetch an entry given its unique identifier

get_parser = <flask_restx.reqparse.RequestParser object>

methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}

A list of methods this view can handle.

put(*id, user=None*)

Update an entry given its unique identifier

```
class metadata_registration_api.api.api_form.ArgsField(default=None, attribute=None, title=None,
                                                         description=None, required=None,
                                                         readonly=None, example=None,
                                                         mask=None, **kwargs)
```

Bases: flask_restx.fields.Raw

Custom field for args field. If the value contains a DataObject, it is modeled with the objects model

format(*value*)

Formats a field's value. No-op by default - field classes that modify how the value of existing object keys should be presented should override this and apply the appropriate formatting.

Parameters **value** – The value to format

Raises **MarshallingError** – In case of formatting problem

Ex:

```
class TitleCase(Raw):
    def format(self, value):
        return unicode(value).title()
```

```
class metadata_registration_api.api.api_form.SubformAddField(default=None, attribute=None,
                                                             title=None, description=None,
                                                             required=None, readonly=None,
                                                             example=None, mask=None,
                                                             **kwargs)
```

Bases: flask_restx.fields.Raw

Custom field for subforms

format(value)

Formats a field's value. No-op by default - field classes that modify how the value of existing object keys should be presented should override this and apply the appropriate formatting.

Parameters value – The value to format

Raises **MarshallingError** – In case of formatting problem

Ex:

```
class TitleCase(Raw):
    def format(self, value):
        return unicode(value).title()
```

```
class metadata_registration_api.api.api_form.SubformField(default=None, attribute=None,
                                                          title=None, description=None,
                                                          required=None, readonly=None,
                                                          example=None, mask=None, **kwargs)
```

Bases: flask_restx.fields.Raw

Custom field for subforms

format(value)

Formats a field's value. No-op by default - field classes that modify how the value of existing object keys should be presented should override this and apply the appropriate formatting.

Parameters value – The value to format

Raises **MarshallingError** – In case of formatting problem

Ex:

```
class TitleCase(Raw):
    def format(self, value):
        return unicode(value).title()
```

metadata_registration_api.api.api_props module

```
class metadata_registration_api.api.api_props.ApiProperties(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(user=None)
        Deprecates all entries

    get()
        Fetch a list with all entries

    get_parser = <flask_restx.reqparse.RequestParser object>

    methods: Optional[List[str]] = {'DELETE', 'GET', 'POST'}
        A list of methods this view can handle.

    post(user=None)
        Add a new entry

        The name has to be unique and is internally used as a variable name. The passed string is preprocessed
        before it is inserted into the database. Preprocessing: All characters are converted to lower case, the leading
        and trailing white spaces are removed, and intermediate white spaces are replaced with underscores (“_”).

        Do not pass a unique identifier since it is generated internally.

        synonyms (optional)

        deprecated (default=False)

        If a data type other than “cv” is added, the controlled_vocabulary is not considered.

class metadata_registration_api.api.api_props.ApiPropertyId(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(id, user=None)
        Deprecates an entry given its unique identifier

    get(id)
        Fetch an entry given its unique identifier

    methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}
        A list of methods this view can handle.

    put(id, user=None)
        Update entry given its unique identifier

metadata_registration_api.api.api_props.check_synonyms_unicity(new_property_payload,
                                                                exclude_id=None)

metadata_registration_api.api.api_props.validate_controlled_vocabulary(entry)
```

metadata_registration_api.api.api_study module

```
class metadata_registration_api.api.api_study.ApiStudy(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(user=None)
        Deprecates all entries

    get(user=None)
        Fetch a list with all entries
```

```

methods: Optional[List[str]] = {'DELETE', 'GET', 'POST'}
    A list of methods this view can handle.

post(user=None)
    Add a new entry

class metadata_registration_api.api.api_study.ApiStudyId(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(id, user=None)
        Delete an entry given its unique identifier

    get(id, user=None)
        Fetch an entry given its unique identifier

    methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}
        A list of methods this view can handle.

    put(id, user=None)
        Update an entry given its unique identifier

metadata_registration_api.api.api_study.check_alternate_pk_unicity(entries, pseudo_apks,
                                                                    prop_map)

    Another dirty mongoDB aggregation to enforce unicity of certain entry properties.

metadata_registration_api.api.api_study.index_study_if_es(study, entries, action)

metadata_registration_api.api.api_study.update_study(study, study_converter, payload, message,
                                                        user=None)

    Steps to update study state, metadata and upload to DB

metadata_registration_api.api.api_study.validate_form_format_against_form(form_name,
                                                                    form_data,
                                                                    form_cls=None)

```

metadata_registration_api.api.api_user module

```

class metadata_registration_api.api.api_user.ApiUser(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(user=None)
        Delete all entries

    get()
        Fetch a list with all entries

    methods: Optional[List[str]] = {'DELETE', 'GET', 'POST'}
        A list of methods this view can handle.

    parser = <flask_restx.reqparse.RequestParser object>

    post(user=None)
        Add a new entry

class metadata_registration_api.api.api_user.ApiUserId(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource

    delete(id, user=None)
        Delete an entry given its unique identifier

    get(id)
        Fetch an entry given its unique identifier

```

methods: `Optional[List[str]] = {'DELETE', 'GET', 'PUT'}`

A list of methods this view can handle.

put(*id*, *user=None*)

Update an entry given its unique identifier

class `metadata_registration_api.api.api_user.Login`(*api=None*, **args*, ***kwargs*)

Bases: `flask_restx.resource.Resource`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post()

Fetch an access token to perform requests which require elevated privileges

Upon successful login, you receive an access token. Pass the token as value of 'x-access-token' in the header of every request that requires elevated privileges. The token is only valid for a certain time interval.

metadata_registration_api.api.api_utils module

class `metadata_registration_api.api.api_utils.ChangeLog`(*action: str*, *user_id: str*, *timestamp: datetime.datetime*, *manual_user: Union[str, NoneType] = None*)

Bases: `object`

action: `str`

manual_user: `Optional[str] = None`

timestamp: `datetime.datetime`

to_dict()

user_id: `str`

class `metadata_registration_api.api.api_utils.MetaInformation`(*state: str*, *change_log=None*)

Bases: `object`

add_log(*log: metadata_registration_api.api.api_utils.ChangeLog*)

to_json()

`metadata_registration_api.api.api_utils.get_cv_items_map`(*key='name'*, *value='label'*)

Returns a map to find the CV item labels in this format: {cv_name: {item_name: item_label}}

`metadata_registration_api.api.api_utils.get_json`(*url*, *headers={}*)

`metadata_registration_api.api.api_utils.get_mask`(*request*)

`metadata_registration_api.api.api_utils.get_property_map`(*key*, *value*, *mask=None*)

Helper to get property mapper

metadata_registration_api.api.decorators module

`metadata_registration_api.api.decorators.token_required(f)`

A decorator to ensure that the request contains an access token

Module contents

`metadata_registration_api.api.general_error_handler(error)`

`metadata_registration_api.api.handle_authorization_error(error)`

`metadata_registration_api.api.handle_does_not_exist_error(error)`

`metadata_registration_api.api.handle_not_unique_error(error)`

`metadata_registration_api.api.http_error_handler(error)`

`metadata_registration_api.api.state_machine_exception(error)`

3.1.2 Submodules

3.1.3 metadata_registration_api.app module

`metadata_registration_api.app.config_app(app)`

`metadata_registration_api.app.create_app()`

3.1.4 metadata_registration_api.errors module

exception `metadata_registration_api.errors.ApiBaseException`

Bases: `Exception`

exception `metadata_registration_api.errors.IdenticalPropertyException`

Bases: `metadata_registration_api.errors.ApiBaseException`

exception `metadata_registration_api.errors.RequestBodyException`

Bases: `metadata_registration_api.errors.ApiBaseException`

exception `metadata_registration_api.errors.TokenException`

Bases: `metadata_registration_api.errors.ApiBaseException`

3.1.5 metadata_registration_api.model module

class `metadata_registration_api.model.ControlledVocabulary(*args, **values)`

Bases: `metadata_registration_api.model.TopLevelDocument`

Model for a controlled vocabulary.

A controlled vocabulary contains a list of possible items. See `Property`.

exception `DoesNotExist`

Bases: `mongoengine.errors.DoesNotExist`

exception `MultipleObjectsReturned`

Bases: `mongoengine.errors.MultipleObjectsReturned`

description

A unicode string field.

id

A field wrapper around MongoDB's ObjectIds.

items

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: many-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

objects

The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet`, probably the same one that was passed in, but modified in some way.

validate(*args, **kwargs)

Enforce unicity of CV item synonyms

class `metadata_registration_api.model.CvItem(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

An item in the list of controlled vocabularies

description

A unicode string field.

label

A unicode string field.

name

A unicode string field.

synonyms

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: many-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

class `metadata_registration_api.model.DataObject(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

args

A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

class_name

A unicode string field.

fields

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: many-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

kwargs

A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is { }

property

A reference to a document that will be automatically dereferenced on access (lazily).

Note this means you will get a database I/O access everytime you access this field. This is necessary because the field returns a `Document` which precise type can depend of the value of the `_cls` field present in the document in database. In short, using this type of field can lead to poor performances (especially if you access this field only to retrieve its `pk` field which is already known before dereference). To solve this you should consider using the `LazyReferenceField`.

Use the `reverse_delete_rule` to handle what should happen if the document the field is referencing is deleted. `EmbeddedDocuments`, `DictFields` and `MapFields` does not support `reverse_delete_rule` and an `InvalidDocumentError` will be raised if trying to set on one of these Document / Field types.

The options are:

- `DO_NOTHING (0)` - don't do anything (default).
- `NULLIFY (1)` - Updates the reference to null.
- `CASCADE (2)` - Deletes the documents associated with the reference.
- `DENY (3)` - Prevent the deletion of the reference object.
- `PULL (4)` - Pull the reference from a `ListField` of references

Alternative syntax for registering delete rules (useful when implementing bi-directional delete rules)

```
class Org(Document):
    owner = ReferenceField('User')

class User(Document):
    org = ReferenceField('Org', reverse_delete_rule=CASCADE)

User.register_delete_rule(Org, 'owner', DENY)
```

class `metadata_registration_api.model.DataObjects(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

objects

A `ListField` designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of `EmbeddedDocument`.

class `metadata_registration_api.model.Form(*args, **values)`

Bases: `metadata_registration_api.model.TopLevelDocument`

MongoDB representation of a FlaskForm

The form contains multiple fields.

exception `DoesNotExist`

Bases: `mongoengine.errors.DoesNotExist`

exception `MultipleObjectsReturned`

Bases: `mongoengine.errors.MultipleObjectsReturned`

description

A unicode string field.

fields

A `ListField` designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of `EmbeddedDocument`.

id

A field wrapper around MongoDB's `ObjectIds`.

objects

The default `QuerySet` Manager.

Custom `QuerySet` Manager functions can extend this class and users can add extra `queryset` functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet`, probably the same one that was passed in, but modified in some way.

class `metadata_registration_api.model.FormField(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

args

A generic embedded document field - allows any `EmbeddedDocument` to be stored.

Only valid values are subclasses of `EmbeddedDocument`.

Note: You can use the `choices` param to limit the acceptable `EmbeddedDocument` types

class_name

A unicode string field.

clean()

Hook for doing document level data cleaning (usually validation or assignment) before validation is run.

Any `ValidationError` raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by `NON_FIELD_ERRORS`.

description

A unicode string field.

fields

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: many-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

kwargs

A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is { }

label

A unicode string field.

name

A unicode string field.

property

A reference to a document that will be automatically dereferenced on access (lazily).

Note this means you will get a database I/O access everytime you access this field. This is necessary because the field returns a `Document` which precise type can depend of the value of the `_cls` field present in the document in database. In short, using this type of field can lead to poor performances (especially if you access this field only to retrieve it `pk` field which is already known before dereference). To solve this you should consider using the `LazyReferenceField`.

Use the `reverse_delete_rule` to handle what should happen if the document the field is referencing is deleted. `EmbeddedDocuments`, `DictFields` and `MapFields` does not support `reverse_delete_rule` and an `InvalidDocumentError` will be raised if trying to set on one of these Document / Field types.

The options are:

- `DO_NOTHING` (0) - don't do anything (default).
- `NULLIFY` (1) - Updates the reference to null.
- `CASCADE` (2) - Deletes the documents associated with the reference.
- `DENY` (3) - Prevent the deletion of the reference object.
- `PULL` (4) - Pull the reference from a `ListField` of references

Alternative syntax for registering delete rules (useful when implementing bi-directional delete rules)

```
class Org(Document):
    owner = ReferenceField('User')

class User(Document):
    org = ReferenceField('Org', reverse_delete_rule=CASCADE)

User.register_delete_rule(Org, 'owner', DENY)
```

```
class metadata_registration_api.model.History(*args, **kwargs)
```

Bases: `mongoengine.document.EmbeddedDocument`

action

A unicode string field.

manual_user

A unicode string field.

timestamp

Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

user_id

A reference to a document that will be automatically dereferenced on access (lazily).

Note this means you will get a database I/O access everytime you access this field. This is necessary because the field returns a Document which precise type can depend of the value of the `_cls` field present in the document in database. In short, using this type of field can lead to poor performances (especially if you access this field only to retrieve it `pk` field which is already known before dereference). To solve this you should consider using the LazyReferenceField.

Use the `reverse_delete_rule` to handle what should happen if the document the field is referencing is deleted. EmbeddedDocuments, DictFields and MapFields does not support `reverse_delete_rule` and an `InvalidDocumentError` will be raised if trying to set on one of these Document / Field types.

The options are:

- DO_NOTHING (0) - don't do anything (default).
- NULLIFY (1) - Updates the reference to null.
- CASCADE (2) - Deletes the documents associated with the reference.
- DENY (3) - Prevent the deletion of the reference object.
- PULL (4) - Pull the reference from a ListField of references

Alternative syntax for registering delete rules (useful when implementing bi-directional delete rules)

```
class Org(Document):
    owner = ReferenceField('User')

class User(Document):
    org = ReferenceField('Org', reverse_delete_rule=CASCADE)

User.register_delete_rule(Org, 'owner', DENY)
```

```
class metadata_registration_api.model.MetaInformation(*args, **kwargs)
```

Bases: mongoengine.document.EmbeddedDocument

change_log

A ListField designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of EmbeddedDocument.

deprecated

Boolean field type.

state

A unicode string field.

class `metadata_registration_api.model.Property(*args, **values)`

Bases: `metadata_registration_api.model.TopLevelDocument`

Model for a property

A property is assigned to a level.

exception DoesNotExist

Bases: `mongoengine.errors.DoesNotExist`

exception MultipleObjectsReturned

Bases: `mongoengine.errors.MultipleObjectsReturned`

clean()

Hook for doing document level data cleaning (usually validation or assignment) before validation is run.

Any `ValidationError` raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by `NON_FIELD_ERRORS`.

description

A unicode string field.

id

A field wrapper around MongoDB's `ObjectIds`.

level

A unicode string field.

objects

The default `QuerySet` Manager.

Custom `QuerySet` Manager functions can extend this class and users can add extra `queryset` functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet`, probably the same one that was passed in, but modified in some way.

synonyms

A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with `ReferenceFields` see: `many-to-many-with-listfields`

Note: Required means it cannot be empty - as the default for `ListFields` is []

value_type

An embedded document field - with a declared `document_type`. Only valid values are subclasses of `EmbeddedDocument`.

class `metadata_registration_api.model.Study(*args, **values)`

Bases: `mongoengine.document.Document`

exception DoesNotExist

Bases: `mongoengine.errors.DoesNotExist`

exception MultipleObjectsReturned

Bases: `mongoengine.errors.MultipleObjectsReturned`

entries

A `ListField` designed specially to hold a list of embedded documents to provide additional query helpers.

Note: The only valid list values are subclasses of `EmbeddedDocument`.

id

A field wrapper around MongoDB's `ObjectIds`.

meta_information

An embedded document field - with a declared `document_type`. Only valid values are subclasses of `EmbeddedDocument`.

objects

The default `QuerySet` Manager.

Custom `QuerySet` Manager functions can extend this class and users can add extra `queryset` functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet`, probably the same one that was passed in, but modified in some way.

class `metadata_registration_api.model.StudyEntry(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

property

A reference to a document that will be automatically dereferenced on access (lazily).

Note this means you will get a database I/O access everytime you access this field. This is necessary because the field returns a `Document` which precise type can depend of the value of the `_cls` field present in the document in database. In short, using this type of field can lead to poor performances (especially if you access this field only to retrieve its `pk` field which is already known before dereference). To solve this you should consider using the `LazyReferenceField`.

Use the `reverse_delete_rule` to handle what should happen if the document the field is referencing is deleted. `EmbeddedDocuments`, `DictFields` and `MapFields` does not support `reverse_delete_rule` and an `InvalidDocumentError` will be raised if trying to set on one of these `Document` / `Field` types.

The options are:

- `DO_NOTHING` (0) - don't do anything (default).
- `NULLIFY` (1) - Updates the reference to null.
- `CASCADE` (2) - Deletes the documents associated with the reference.
- `DENY` (3) - Prevent the deletion of the reference object.
- `PULL` (4) - Pull the reference from a `ListField` of references

Alternative syntax for registering delete rules (useful when implementing bi-directional delete rules)

```
class Org(Document):
    owner = ReferenceField('User')

class User(Document):
    org = ReferenceField('Org', reverse_delete_rule=CASCADE)
```

(continues on next page)

(continued from previous page)

```
User.register_delete_rule(Org, 'owner', DENY)
```

value

A truly dynamic field type capable of handling different and varying types of data.

Used by `DynamicDocument` to handle dynamic data

class `metadata_registration_api.model.TopLevelDocument(*args, **values)`

Bases: `mongoengine.document.Document`

Base class for all top level documents

All top level documents have a *label*, a *name* and a *deprecated* flag. The *label* is for displaying to the end user (external representation) and the *name* is used by the machine (internal representation). The name is expected to be unique for the model. To ensure that the *name* is converted to snake case before it is inserted into the database. The *deprecated* flag indicates if a document is no longer needed (alternative to delete it)

clean()

Hook for doing document level data cleaning (usually validation or assignment) before validation is run.

Any `ValidationError` raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by `NON_FIELD_ERRORS`.

deprecated

Boolean field type.

label

A unicode string field.

name

A unicode string field.

class `metadata_registration_api.model.User(*args, **values)`

Bases: `mongoengine.document.Document`

exception DoesNotExist

Bases: `mongoengine.errors.DoesNotExist`

exception MultipleObjectsReturned

Bases: `mongoengine.errors.MultipleObjectsReturned`

clean()

Called before data is inserted into the database

email

A field that validates input as an email address.

firstname

A unicode string field.

id

A field wrapper around MongoDB's `ObjectIds`.

is_active

Boolean field type.

lastname

A unicode string field.

objects

The default `QuerySet` Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet`, probably the same one that was passed in, but modified in some way.

password

A unicode string field.

class `metadata_registration_api.model.VocabularyType(*args, **kwargs)`

Bases: `mongoengine.document.EmbeddedDocument`

Model which defines the allowed vocabulary.

It is used to validate user input. If the data type is `ctrl_voc`, only the items of `ControlledVocabulary` are allowed.

controlled_vocabulary

A reference to a document that will be automatically dereferenced on access (lazily).

Note this means you will get a database I/O access everytime you access this field. This is necessary because the field returns a `Document` which precise type can depend of the value of the `_cls` field present in the document in database. In short, using this type of field can lead to poor performances (especially if you access this field only to retrieve its `pk` field which is already known before dereference). To solve this you should consider using the `LazyReferenceField`.

Use the `reverse_delete_rule` to handle what should happen if the document the field is referencing is deleted. `EmbeddedDocuments`, `DictFields` and `MapFields` does not support `reverse_delete_rule` and an `InvalidDocumentError` will be raised if trying to set on one of these `Document` / `Field` types.

The options are:

- `DO_NOTHING (0)` - don't do anything (default).
- `NULLIFY (1)` - Updates the reference to null.
- `CASCADE (2)` - Deletes the documents associated with the reference.
- `DENY (3)` - Prevent the deletion of the reference object.
- `PULL (4)` - Pull the reference from a `ListField` of references

Alternative syntax for registering delete rules (useful when implementing bi-directional delete rules)

```
class Org(Document):
    owner = ReferenceField('User')

class User(Document):
    org = ReferenceField('Org', reverse_delete_rule=CASCADE)

User.register_delete_rule(Org, 'owner', DENY)
```

data_type

A unicode string field.

`metadata_registration_api.model.to_snake_case(name)`

Convert a string into an internal representation (no leading and trailing whitespace, and intermediate whitespace replaced with underscore)

Parameters `name` – a given name

Returns `name` in `snake_case` or `None`

3.1.6 metadata_registration_api.my_utils module

3.1.7 Module contents

3.2 scripts package

3.2.1 Submodules

3.2.2 scripts.rna_seq_upload module

3.2.3 scripts.setup module

3.2.4 Module contents

3.3 test package

3.3.1 Submodules

3.3.2 test.test_api_base module

```
class test.test_api_base.BaseTestCase(methodName='runTest')
    Bases: unittest.case.TestCase
    Run the WSGI server in separate daemon thread
    classmethod setUpClass() → None
    classmethod tearDownClass() → None
```

3.3.3 test.test_api_cv module

3.3.4 test.test_api_form module

3.3.5 test.test_api_property module

3.3.6 test.test_api_study module

3.3.7 test.test_api_user module

3.3.8 test.test_api_util module

```
class test.test_api_util.TestAPIUtil(methodName='runTest')
    Bases: unittest.case.TestCase
    test_meta_information_empty_change_log()
```

3.3.9 test.test_main module

`test.test_main.suite()`

3.3.10 test.test_utils module

Utility function used for only testing

class `test.test_utils.ServerThread`(*config='TESTING'*)

Bases: `threading.Thread`

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard `run()` method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the `args` and `kwargs` arguments, respectively.

shutdown()

`test.test_utils.insert`(*url=None, data=None*)

Insert a new entry

3.3.11 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

t

test, [24](#)

test.test_api_base, [23](#)

test.test_api_util, [23](#)

test.test_main, [24](#)

test.test_utils, [24](#)

INDEX

A

`action` (`metadata_registration_api.api.api_utils.ChangeLog` attribute), 12
`action` (`metadata_registration_api.model.History` attribute), 17
`add_log()` (`metadata_registration_api.api.api_utils.MetadataInformation` method), 12
`ApiBaseException`, 13
`ApiControlledVocabulary` (class in `metadata_registration_api.api.api_ctrl_voc`), 7
`ApiControlledVocabularyId` (class in `metadata_registration_api.api.api_ctrl_voc`), 7
`ApiControlledVocabularyItemsMap` (class in `metadata_registration_api.api.api_ctrl_voc`), 7
`ApiForm` (class in `metadata_registration_api.api.api_form`), 8
`ApiFormId` (class in `metadata_registration_api.api.api_form`), 8
`ApiProperties` (class in `metadata_registration_api.api.api_props`), 10
`ApiPropertyId` (class in `metadata_registration_api.api.api_props`), 10
`ApiStudy` (class in `metadata_registration_api.api.api_study`), 10
`ApiStudyId` (class in `metadata_registration_api.api.api_study`), 11
`ApiUser` (class in `metadata_registration_api.api.api_user`), 11
`ApiUserId` (class in `metadata_registration_api.api.api_user`), 11
`args` (`metadata_registration_api.model.DataObject` attribute), 14
`args` (`metadata_registration_api.model.FormField` attribute), 16
`ArgsField` (class in `metadata_registration_api.api.api_form`), 8

B

`BaseTestCase` (class in `test.test_api_base`), 23

C

`change_log` (`metadata_registration_api.model.MetadataInformation` attribute), 18
`ChangeLog` (class in `metadata_registration_api.api.api_utils`), 12
`check_alternate_pk_unicity()` (in module `metadata_registration_api.api.api_study`), 11
`check_synonyms_unicity()` (in module `metadata_registration_api.api.api_props`), 10
`class_name` (`metadata_registration_api.model.DataObject` attribute), 14
`class_name` (`metadata_registration_api.model.FormField` attribute), 16
`clean()` (`metadata_registration_api.model.FormField` method), 16
`clean()` (`metadata_registration_api.model.Property` method), 19
`clean()` (`metadata_registration_api.model.TopLevelDocument` method), 21
`clean()` (`metadata_registration_api.model.User` method), 21
`config_app()` (in module `metadata_registration_api.app`), 13
`controlled_vocabulary` (`metadata_registration_api.model.VocabularyType` attribute), 22
`ControlledVocabulary` (class in `metadata_registration_api.model`), 13
`ControlledVocabulary.DoesNotExist`, 13
`ControlledVocabulary.MultipleObjectsReturned`, 13
`create_app()` (in module `metadata_registration_api.app`), 13
`CvItem` (class in `metadata_registration_api.model`), 14

D

`data_type` (`metadata_registration_api.model.VocabularyType` attribute), 22
`DataObject` (class in `metadata_registration_api.model`), 14
`DataObjects` (class in `metadata_registration_api.model`), 15

[delete\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary` attribute), 7
[delete\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiFormId` attribute), 8
[delete\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiFormId` attribute), 9
[delete\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 8
[delete\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 9
[delete\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 16
[delete\(\)](#) (`metadata_registration_api.api.api_props.ApiProperties` attribute), 10
[delete\(\)](#) (`metadata_registration_api.api.api_props.ApiProperties` attribute), 10
[delete\(\)](#) (`metadata_registration_api.api.api_study.ApiStudyId` attribute), 10
[delete\(\)](#) (`metadata_registration_api.api.api_study.ApiStudyId` attribute), 11
[delete\(\)](#) (`metadata_registration_api.api.api_user.ApiUserId` attribute), 11
[delete\(\)](#) (`metadata_registration_api.api.api_user.ApiUserId` attribute), 11
[deprecated](#) (`metadata_registration_api.model.MetaInformation` attribute), 18
[deprecated](#) (`metadata_registration_api.model.TopLevelDomain` attribute), 21
[description](#) (`metadata_registration_api.model.ControlledVocabulary` attribute), 13
[description](#) (`metadata_registration_api.model.CvItem` attribute), 14
[description](#) (`metadata_registration_api.model.Form` attribute), 16
[description](#) (`metadata_registration_api.model.FormField` attribute), 16
[description](#) (`metadata_registration_api.model.Property` attribute), 19
E
[email](#) (`metadata_registration_api.model.User` attribute), 21
[entries](#) (`metadata_registration_api.model.Study` attribute), 20
F
[fields](#) (`metadata_registration_api.model.DataObject` attribute), 15
[fields](#) (`metadata_registration_api.model.Form` attribute), 16
[fields](#) (`metadata_registration_api.model.FormField` attribute), 16
[firstname](#) (`metadata_registration_api.model.User` attribute), 21
[Form](#) (`class in metadata_registration_api.model`), 16
[Form.DoesNotExist](#), 16
[Form.MultipleObjectsReturned](#), 16
[format\(\)](#) (`metadata_registration_api.api.api_form.ArgsField` attribute), 8
[format\(\)](#) (`metadata_registration_api.api.api_form.SubformAddField` attribute), 9
[format\(\)](#) (`metadata_registration_api.api.api_form.SubformField` attribute), 9
[FormField](#) (`class in metadata_registration_api.model`), 16
G
[get_cv_items_map\(\)](#) (`in module metadata_registration_api.api`), 13
[get\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary` attribute), 7
[get\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary` attribute), 7
[get\(\)](#) (`metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary` attribute), 8
[get\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 8
[get\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 8
[get\(\)](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 10
[get\(\)](#) (`metadata_registration_api.api.api_props.ApiProperties` attribute), 10
[get\(\)](#) (`metadata_registration_api.api.api_props.ApiPropertyId` attribute), 10
[get\(\)](#) (`metadata_registration_api.api.api_study.ApiStudyId` attribute), 10
[get\(\)](#) (`metadata_registration_api.api.api_study.ApiStudyId` attribute), 11
[get\(\)](#) (`metadata_registration_api.api.api_user.ApiUserId` attribute), 11
[get\(\)](#) (`metadata_registration_api.api.api_user.ApiUserId` attribute), 11
[get_cv_items_map\(\)](#) (`in module metadata_registration_api.api.api_utils`), 12
[get_json\(\)](#) (`in module metadata_registration_api.api.api_utils`), 12
[get_mask\(\)](#) (`in module metadata_registration_api.api.api_utils`), 12
[get_parser](#) (`metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary` attribute), 7
[get_parser](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 8
[get_parser](#) (`metadata_registration_api.api.api_form.ApiFormId` attribute), 8
[get_parser](#) (`metadata_registration_api.api.api_props.ApiProperties` attribute), 10
[get_property_map\(\)](#) (`in module metadata_registration_api.api.api_utils`), 12
H
[handle_authorization_error\(\)](#) (`in module metadata_registration_api`), 13

[handle_does_not_exist_error\(\)](#) (in module *metadata_registration_api.api*), 13
[handle_not_unique_error\(\)](#) (in module *metadata_registration_api.api*), 13
[History](#) (class in *metadata_registration_api.model*), 17
[http_error_handler\(\)](#) (in module *metadata_registration_api.api*), 13
I
[id](#) (*metadata_registration_api.model.ControlledVocabulary* attribute), 14
[id](#) (*metadata_registration_api.model.Form* attribute), 16
[id](#) (*metadata_registration_api.model.Property* attribute), 19
[id](#) (*metadata_registration_api.model.Study* attribute), 20
[id](#) (*metadata_registration_api.model.User* attribute), 21
[IdenticalPropertyException](#), 13
[index_study_if_es\(\)](#) (in module *metadata_registration_api.api.api_study*), 11
[insert\(\)](#) (in module *test.test_utils*), 24
[is_active](#) (*metadata_registration_api.model.User* attribute), 21
[items](#) (*metadata_registration_api.model.ControlledVocabulary* attribute), 14
K
[kwargs](#) (*metadata_registration_api.model.DataObject* attribute), 15
[kwargs](#) (*metadata_registration_api.model.FormField* attribute), 17
L
[label](#) (*metadata_registration_api.model.CvItem* attribute), 14
[label](#) (*metadata_registration_api.model.FormField* attribute), 17
[label](#) (*metadata_registration_api.model.TopLevelDocument* attribute), 21
[lastname](#) (*metadata_registration_api.model.User* attribute), 21
[level](#) (*metadata_registration_api.model.Property* attribute), 19
[Login](#) (class in *metadata_registration_api.api.api_user*), 12
M
[manual_user](#) (*metadata_registration_api.api.api_utils.ChangeLog* attribute), 12
[manual_user](#) (*metadata_registration_api.model.History* attribute), 18
[meta_information](#) (*metadata_registration_api.model.Study* attribute), 20
[metadata_registration_api](#)

- module, 23
- metadata_registration_api.api*
 - module, 13
 - metadata_registration_api.api.api_ctrl_voc*
 - module, 7
 - metadata_registration_api.api.api_form*
 - module, 8
 - metadata_registration_api.api.api_props*
 - module, 10
 - metadata_registration_api.api.api_study*
 - module, 10
 - metadata_registration_api.api.api_user*
 - module, 11
 - metadata_registration_api.api.api_utils*
 - module, 12
 - metadata_registration_api.api.decorators*
 - module, 13
 - metadata_registration_api.app*
 - module, 13
 - metadata_registration_api.errors*
 - module, 13
 - metadata_registration_api.model*
 - module, 13
 - MetaInformation* (class in *metadata_registration_api.api.api_utils*), 12
 - MetaInformation* (class in *metadata_registration_api.model*), 18
 - methods* (*metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary* attribute), 7
 - methods* (*metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary* attribute), 7
 - methods* (*metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary* attribute), 8
 - methods* (*metadata_registration_api.api.api_form.ApiForm* attribute), 8
 - methods* (*metadata_registration_api.api.api_form.ApiFormId* attribute), 8
 - methods* (*metadata_registration_api.api.api_props.ApiProperties* attribute), 10
 - methods* (*metadata_registration_api.api.api_props.ApiPropertyId* attribute), 10
 - methods* (*metadata_registration_api.api.api_study.ApiStudy* attribute), 10
 - methods* (*metadata_registration_api.api.api_study.ApiStudyId* attribute), 11
 - methods* (*metadata_registration_api.api.api_user.ApiUser* attribute), 11
 - methods* (*metadata_registration_api.api.api_user.ApiUserId* attribute), 11
 - methods* (*metadata_registration_api.api.api_user.Login* attribute), 12
 - module
 - metadata_registration_api*, 23
 - metadata_registration_api.api*, 13

[metadata_registration_api.api.api_ctrl_voc.post\(\)](#) ([metadata_registration_api.api.api_props.ApiProperties](#) method), 10
[metadata_registration_api.api.api_form](#), 8
[metadata_registration_api.api.api_props](#), 10
[metadata_registration_api.api.api_study](#), 10
[metadata_registration_api.api.api_user](#), 11
[metadata_registration_api.api.api_utils](#), 12
[metadata_registration_api.api.decorators](#), 13
[metadata_registration_api.app](#), 13
[metadata_registration_api.errors](#), 13
[metadata_registration_api.model](#), 13
[test](#), 24
[test.test_api_base](#), 23
[test.test_api_util](#), 23
[test.test_main](#), 24
[test.test_utils](#), 24

N

[name](#) ([metadata_registration_api.model.CvItem](#) attribute), 14
[name](#) ([metadata_registration_api.model.FormField](#) attribute), 17
[name](#) ([metadata_registration_api.model.TopLevelDocument](#) attribute), 21

O

[objects](#) ([metadata_registration_api.model.ControlledVocabulary](#) attribute), 14
[objects](#) ([metadata_registration_api.model.DataObjects](#) attribute), 15
[objects](#) ([metadata_registration_api.model.Form](#) attribute), 16
[objects](#) ([metadata_registration_api.model.Property](#) attribute), 19
[objects](#) ([metadata_registration_api.model.Study](#) attribute), 20
[objects](#) ([metadata_registration_api.model.User](#) attribute), 21

P

[parser](#) ([metadata_registration_api.api.api_user.ApiUser](#) attribute), 11
[password](#) ([metadata_registration_api.model.User](#) attribute), 22
[post\(\)](#) ([metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary](#) method), 7
[post\(\)](#) ([metadata_registration_api.api.api_form.ApiForm](#) method), 8

[post\(\)](#) ([metadata_registration_api.api.api_study.ApiStudy](#) method), 11
[post\(\)](#) ([metadata_registration_api.api.api_user.ApiUser](#) method), 11
[post\(\)](#) ([metadata_registration_api.api.api_user.Login](#) method), 12
[Property](#) (class in [metadata_registration_api.model](#)), 19
[property](#) ([metadata_registration_api.model.DataObject](#) attribute), 15
[property](#) ([metadata_registration_api.model.FormField](#) attribute), 17
[property](#) ([metadata_registration_api.model.StudyEntry](#) attribute), 20
[Property.DoesNotExist](#), 19
[Property.MultipleObjectsReturned](#), 19
[put\(\)](#) ([metadata_registration_api.api.api_ctrl_voc.ApiControlledVocabulary](#) method), 7
[put\(\)](#) ([metadata_registration_api.api.api_form.ApiFormId](#) method), 8
[put\(\)](#) ([metadata_registration_api.api.api_props.ApiPropertyId](#) method), 10
[put\(\)](#) ([metadata_registration_api.api.api_study.ApiStudyId](#) method), 11
[put\(\)](#) ([metadata_registration_api.api.api_user.ApiUserId](#) method), 12

R

[RequestBodyException](#), 13
[run\(\)](#) ([test.test_utils.ServerThread](#) method), 24

S

[ServerThread](#) (class in [test.test_utils](#)), 24
[setUpClass\(\)](#) ([test.test_api_base.BaseTestCase](#) class method), 23
[shutdown\(\)](#) ([test.test_utils.ServerThread](#) method), 24
[state](#) ([metadata_registration_api.model.MetaInformation](#) attribute), 19
[state_machine_exception\(\)](#) (in module [metadata_registration_api](#)), 13
[Study](#) (class in [metadata_registration_api.model](#)), 19
[Study.DoesNotExist](#), 19
[Study.MultipleObjectsReturned](#), 19
[StudyEntry](#) (class in [metadata_registration_api.model](#)), 20
[SubformAddField](#) (class in [metadata_registration_api.api.api_form](#)), 9
[SubFormField](#) (class in [metadata_registration_api.api.api_form](#)), 9
[suite\(\)](#) (in module [test.test_main](#)), 24
[synonyms](#) ([metadata_registration_api.model.CvItem](#) attribute), 14

`synonyms` (*metadata_registration_api.model.Property* attribute), 19

T

`tearDownClass()` (*test.test_api_base.BaseTestCase* class method), 23

`test`

`module`, 24

`test.test_api_base`

`module`, 23

`test.test_api_util`

`module`, 23

`test.test_main`

`module`, 24

`test.test_utils`

`module`, 24

`test_meta_information_empty_change_log()`
(*test.test_api_util.TestAPIUtil* method), 23

`TestAPIUtil` (class in *test.test_api_util*), 23

`timestamp` (*metadata_registration_api.api.api_utils.ChangeLog* attribute), 12

`timestamp` (*metadata_registration_api.model.History* attribute), 18

`to_dict()` (*metadata_registration_api.api.api_utils.ChangeLog* method), 12

`to_json()` (*metadata_registration_api.api.api_utils.MetaInformation* method), 12

`to_snake_case()` (in *module metadata_registration_api.model*), 22

`token_required()` (in *module metadata_registration_api.api.decorators*), 13

`TokenException`, 13

`TopLevelDocument` (class in *metadata_registration_api.model*), 21

U

`update_study()` (in *module metadata_registration_api.api.api_study*), 11

`User` (class in *metadata_registration_api.model*), 21

`User.DoesNotExist`, 21

`User.MultipleObjectsReturned`, 21

`user_id` (*metadata_registration_api.api.api_utils.ChangeLog* attribute), 12

`user_id` (*metadata_registration_api.model.History* attribute), 18

V

`validate()` (*metadata_registration_api.model.ControlledVocabulary* method), 14

`validate_controlled_vocabulary()` (in *module metadata_registration_api.api.api_props*), 10

`validate_form_format_against_form()` (in *module metadata_registration_api.api.api_study*), 11