# yadm

***Release 1.0.0***

November 14, 2015

Contents

**1 Quick start** **3**
1.1 CHANGES . . . . . 4

**2 API documentation** **5**
2.1 API . . . . . 5

**Python Module Index** **23**

It's small and simple ODM for use with MongoDB.

CHAPTER 1

# Quick start

```python
import pymongo
from yadm import Database, Document, fields


# Create model
class BlogPost(Document):
    __collection__ = 'blog_posts'

    title = fields.StringField()
    body = fields.StringField()


# Create post
post = BlogPost()
post.title = 'Small post'
post.body = 'Bla-bla-bla...'

# Connect to database
client = pymongo.MongoClient("localhost", 27017)
db = Database(client, 'test')

# Insert post to database
db.insert(post)

# Query posts
qs = db.get_queryset(BlogPost).find({'title': {'$regex': '^s.*'}})
assert qs.count() > 0

for post in qs:
    assert post.title.startswith('s')

# Query one post
post = db.get_queryset(BlogPost).find_one({'title': 'Small post'})

# Change post
post.title = 'Bla-bla-bla title'

# Save changed post
db.save(post)
```

## 1.1 CHANGES

### 1.1.1 1.0 (2015-11-14)

- **Change document structure. No more bad *BaseDocument.__data__* attribute:**
  - *BaseDocument.__raw__*: raw data from mongo;
  - *BaseDocument.__cache__*: cached objects, casted with fields;
  - *BaseDocument.__changed__*: changed objects.
- **Changes api for custom fields:**
  - Not more need create field descriptors for every field;
  - *prepare_value* called only for setattr;
  - *to_mongo* called only for save objects to mongo;
  - *from_mongo* called only for load values from *BaseDocument.__raw__*;
  - Remove *Field.default* attribute. Use *Field.get_default* method;
  - Add *get_if_not_loaded* and *get_if_attribute_not_set* method;
  - By default raise *NotLoadedError* if field not loaded from projection;
- **Changes in *ReferenceField*:**
  - Raise *BrokenReference* if link is bloken;
  - Raise *NotBindingToDatabase* if document not saved to database;
- *smart_null* keyword for *Field*;
- Fields in document must be instances (not classes!);
- Remove *ArrayContainer* and *ArrayContainerField*;
- Remove old *MapIntKeysField* and *MapObjectIdKeysField*. Use new *MapCustomKeysField*;
- Add *Database.update_one* method for run simple update query with specified document;
- Add *QuerySet.distinct*;
- *serialize.from_mongo* now accept *not_loaded* sequence with filed names who must mark as not loaded, *parent* and *name*;
- *serialize.to_mongo* do not call *FieldDescriptor.__set__*;
- Fakers! Subsystem for generate test objects;
- Tests now use pytest;
- And more, and more...

CHAPTER 2

# API documentation

## 2.1 API

API documentation

### 2.1.1 Database

This module for provide work with MongoDB database.

```
import pymongo
from yadm.database import Database

from mydocs import Doc

client = pymongo.MongoClient("localhost", 27017)
db = Database(self.client, 'test')

doc = Doc()
db.insert(doc)

doc.arg = 13
db.save(doc)

qs = db.get_queryset(Doc).find({'arg': {'$gt': 10}})
for doc in qs:
    print(doc)
```

**class** yadm.database.**Database**(*client*, *name*)
Main object who provide work with database

**Parameters**

- **client** (*pymongo.Client*) – database connection
- **name** (*str*) – database name

**bulk**(*document_class*, *ordered=False*, *raise_on_errors=True*)
Return Bulk

**Parameters**

- **document_class** (MetaDocument) – class of documents fo bulk
- **ordered** (*bool*) – create ordered bulk (default *False*)

- **raise_on_errors** (*bool*) – raise BulkWriteError exception if write errors (default *True*)

Context manager:

**with db.bulk(Doc) as bulk:** bulk.insert(doc_1) bulk.insert(doc_2)

**get_queryset** (*document_class*)
Return queryset for document class

**Parameters** **document_class** – *yadm.documents.Document*

This create instance of *yadm.queryset.QuerySet* with presetted document's collection information.

**insert** (*document*)
Insert document to database

**Parameters** **document** (Document) – document instance for insert to database

It's bind new document to database set *_id*.

**reload** (*document*, *new_instance=False*)
Reload document

**Parameters**

- **document** (Document) – instance for reload
- **new_instance** (*bool*) – if *True* return new instance of document, else change data in given document (default: *False*)

**remove** (*document*)
Remove document from database

**Parameters** **document** (Document) – instance for remove from database

**save** (*document*, *full=False*, *upsert=False*)
Save document to database

**Parameters**

- **document** (Document) – document instance for save
- **full** (*bool*) – fully resave document (default: *False*)
- **upsert** (*bool*) – see documentation for MongoDB's *update* (default: *False*)

If document has no *_id* *insert* new document.

**update_one** (*document*, *reload=True*, *, *set=None*, *unset=None*, *inc=None*, *push=None*, *pull=None*)
Update one document

**Parameters**

- **document** (Document) – document instance for update
- **reload** (*bool*) – if True, reload document

### 2.1.2 Documents

Basic documents classes for build models.

```
class User(Document):
    __collection__ = 'users'

    first_name = fields.StringField()
    last_name = fields.StringField()
    age = fields.IntegerField()
```

All fields placed in *yadm.fields* package.

**class** yadm.documents.**MetaDocument**(*cls*, *name*, *bases*, *cls_dict*)
Metaclass for documents

**class** yadm.documents.**BaseDocument**(***kwargs*)
Base class for all documents

**__raw__**
Dict with raw data from mongo

**__cache__**
Dict with cached objects, casted with fields

**__changed__**
Dict with changed objects

**__data__**
Deprecated! For backward compatibility only!

Old way to storing data in documents. Now equal to *__raw__*.

**__debug_print__**()
Print debug information

**__fake__**(*values*, *faker*, *depth*)
Fake data customizer

**class** yadm.documents.**Document**(***kwargs*)
Class for build first level documents

**__collection__**
Name of MongoDB collection

**_id**
Mongo object id (bson.ObjectId)

**id**
Alias for *_id* for simply use

**__db__**
Internal attribute contain instance of *yadm.database.Database* for realize yadm.fields.references.ReferenceField. It bind in *yadm.database.Database* or *yadm.queryset.QuerySet*.

**class** yadm.documents.**DocumentItemMixin**
Mixin for custom all fields values, such as *EmbeddedDocument*, *yadm.fields.containers.Container*

**__parent__**
Parent object.

```
        assert doc.embedded_doc.__parent__ is doc
        assert doc.list[13].__parent__ is doc.list
```

**__name__**

```
assert doc.list.__name__ == 'list'
assert doc.list[13].__name__ == 13
```

**__db__**
Database object

```
assert doc.f.l[0].__db__ is doc.__db__
```

**__document__**
Root document

```
assert doc.f.l[0].__document__ is doc
```

**__field_name__**
Dotted field name for MongoDB opperations, like as $set, $push and other...

```
assert doc.f.l[0].__field_name__ == 'f.l.0'
```

**__get_value__**(*document*)
Get value from document with path to self

**__path__**
Path to root generator

```
assert list(doc.f.l[0].__path__) == [doc.f.l[0], doc.f.l, doc.f]
```

**__path_names__**
Path to root generator

```
assert list(doc.f.l[0].__path__) == [0, 'l', 'f']
```

**__weakref__**
list of weak references to the object (if defined)

**class** yadm.documents.**EmbeddedDocument**(***kwargs*)
Class for build embedded documents

### 2.1.3 Serializers and deserializers

Functions for serialize and deserialize data.

yadm.serialize.**from_mongo**(*document_class*, *data*, *not_loaded=()*, *parent=None*, *name=None*)
Deserialize MongoDB data to document

**Parameters**

- **document_class** – document class
- **data** (*dict*) – data from MongoDB
- **not_loaded** (list) – fields, who marked as not loaded
- **parent** – parent for new document
- **name** (*str*) – name for new document

yadm.serialize.**to_mongo**(*document*, *exclude=()*, *include=None*)
Serialize document to MongoDB data

**Parameters**

- **document** (BaseDocument) – document for serializing
- **exclude** (list) – exclude fields
- **include** (list) – include only fields (all by default)

### 2.1.4 Queryset

**class** yadm.queryset.**QuerySet**(*db*, *document_class*)

Query builder

**bulk**()

Return map {id: object}

**Returns** dict

**copy**(**args*, ***kwargs*)

Copy queryset and update it

**Parameters**

- **criteria** (*dict*) –
- **projection** (*dict*) –
- **sort** (*dict*) –

**Returns** new *yadm.queryset.QuerySet* object

**count**()

Count documents in queryset

**Returns** int

**distinct**(*field*)

Distinct query

**Parameters** **field** (*str*) – field for distinct

**Returns** list with result data

**fields**(**fields*)

Get only setted fields

Update projection with fields.

**Parameters** **fields** (*str*) –

**Returns** new *yadm.queryset.QuerySet*

```
qs('field', 'field2')
```

**fields_all**()

Clear projection

**find**(*criteria=None*, *projection=None*)

Return queryset copy with new criteria and projection

**Parameters**

- **criteria** (*dict*) – update queryset's criteria
- **projection** (*dict*) – update queryset's projection

**Returns** new *yadm.queryset.QuerySet*

```
qs({'field': {'$gt': 3}}, {'field': True})
```

**find_and_modify**(*update=None*, *upsert=False*, *full_response=False*, *new=False*, ***kwargs*)
Execute *$findAndModify* query

**Parameters**

- **update** (*dict*) – see second argument to update()
- **upsert** (*bool*) – insert if object doesn't exist *(default False)*
- **full_response** (*bool*) – return the entire response object from the server *(default False)*
- **new** – return updated rather than original object *(default False)*
- **kwargs** – any other options the findAndModify command supports can be passed here

**Returns** *yadm.documents.Document* or **None**

**find_one**(*criteria=None*, *projection=None*)
Find and return only one document

**Parameters**

- **criteria** (*dict*) – update queryset's criteria
- **projection** (*dict*) – update queryset's projection

**Returns** *yadm.documents.Document* or **None**

```
qs({'field': {'$gt': 3}}, {'field': True})
```

**join**(**field_names*)
Create *yadm.Join* object, join *field_names* and return it

**Parameters** **fiels_names** (*str*) – fields for join

**Returns** new *yadm.join.Join*

**Next algorithm for join:**

1. Get all documents from queryset;
2. Aggegate all ids from requested fields;
3. Make *$in* queries for get joined documents;
4. Bind joined documents to objects from first queryset;

*Join* object is instance of *abc.Sequence*.

**remove**()
Remove documents in queryset

**sort**(**sort*)
Sort query

**Parameters** **sort** (*tuples*) – tuples with two items: *('field_name', sort_order_as_int)*.

```
qs.sort(('field_1', 1), ('field_2', -1))
```

**update**(*update*, *multi=True*)
Update documents in queryset

**Parameters**

- **update** (*dict*) – update query
- **multi** (*bool*) – update all matched documents *(default True)*

**Returns** update result

**with_id**(*_id*)
Find document with id

**Parameters** **_id** – id of searching document

**Returns** *yadm.documents.Document* or **None**

### 2.1.5 Bulk queries

**class** yadm.bulk.**Bulk**(*db*, *document_class*, *ordered=False*, *raise_on_errors=True*)
Bulk object

**Parameters**

- **db** (Database) – Database instance
- **document_class** (MetaDocument) – document class for collection
- **ordered** (*bool*) – create ordered bulk (default *False*)
- **raise_on_errors** (*bool*) – raise BulkWriteError exception if write errors (default *True*)

Context manager example:

**with db.bulk(Doc) as bulk:** bulk.insert(doc_1) bulk.insert(doc_2)

**execute**()
Execute the bulk query

**Returns** *BulkResult* instance

**insert**(*document*)
Add insert document to bulk

**Parameters** **document** (Document) – document for insert

> **Warning:** This unlike *Database.insert*! Currently, it is not bind objects to database and set id.

**class** yadm.bulk.**BulkResult**(*bulk*, *raw*)
Object who provide result of *Bulk.execute()*

**n_inserted**
Provide *nInserted* from raw result

**n_modified**
Provide *nModified* from raw result

**n_removed**
Provide *nRemoved* from raw result

**n_upserted**
Provide *nUpserted* from raw result

**write_errors**
Provide *writeErrors* from raw result

### 2.1.6 Join

**class** yadm.join.**Join**(*qs*)
Helper for build client-side joins

```
# Doc.ref is instance of ReferenceField
qs = db(Doc).find({'k': 1})  # queryset filter
join = qs.join('ref')  # create join query in this place
for doc in join:
    print(doc.ref)  # do not create query to database
```

**get_queryset**(*field_name*)
Return queryset for joined objects

**join**(**field_names*)
Do manual join

### 2.1.7 Fields

This package contain all fields.

#### Base fields

Base classes for build database fields.

**class** yadm.fields.base.**NotLoadedError**
Raise if value marked as not loaded

```
doc = db(Doc).fields('a').find_one()
try:
    doc.b
except NotLoadedError:
    print("raised!")
```

**class** yadm.fields.base.**FieldDescriptor**(*name*, *field*)
Base desctiptor for fields

**name**

Name of field

**field**

Field instance for this desctiptor

**__delete__**(*instance*)
Mark document's key as not set

**__get__**(*instance*, *owner*)
Get python value from document

1.Lookup in __changed__;

2.Lookup in __cache__;

3.Lookup in __raw__:

- if AttributeNotSet – call Field.get_if_attribute_not_set;
- if NotLoaded – call Field.get_if_not_loaded;

- call Field.from_mongo;
- set __name__ and __parent__
- save to __cache__

4. Call Field.get_default;
5. If AttributeNotSet – call Field.get_if_attribute_not_set;
6. Return value.

**__set__**(*instance*, *value*)
Set value to document

1. Call Field.prepare_value for cast value;
2. Save in Document.__changed__;
3. Call Field.set_parent_changed.

**class** yadm.fields.base.**Field**(*smart_null=False*)
Base field for all database fields

**Parameters** **smart_null** (*bool*) – If it *True*, access to not exists fields return *None* instead *AttributeError* exception. You will not be able to distinguish null value from not exist. Use with care.

**descriptor_class**
Class of desctiptor for work with field

**document_class**
Class of document. Set in *contribute_to_class()*.

**name**
Name of field in document. Set in *contribute_to_class()*.

**contribute_to_class**(*document_class*, *name*)
Add field for document_class

**Parameters** **document_class** (MetaDocument) – document class for add

**copy**()
Return copy of field

**descriptor_class**
alias of *FieldDescriptor*

**from_mongo**(*document*, *value*)
Convert mongo value to python value

**Parameters**

- **document** (BaseDocument) – document
- **value** – mongo value

**Returns** python value

**get_default**(*document*)
Return default value

**get_fake**(*document*, *faker*, *deep*)
Return fake data for testing

**get_if_attribute_not_set**(*document*)
Call if key not exist in document

**get_if_not_loaded**(*document*)
Call if field data marked as not loaded

**prepare_value**(*document*, *value*)

**The method is called when value is assigned** for the attribute

**Parameters**

- **document** (BaseDocument) – document
- **value** – raw value

**Returns** prepared value

It must be accept *value* argument and return processed (e.g. casted) analog. Also it is called once for the default value.

**to_mongo**(*document*, *value*)
Convert python value to mongo value

**Parameters**

- **document** (BaseDocument) – document
- **value** – python value

**Returns** mongo value

### Simple fields

Fields for basic data types.

**class** yadm.fields.simple.**BooleanField**(*default=<class 'yadm.markers.AttributeNotSet'>*, ***, *choices=None*, ***kwargs*)
Field for boolean values

**type**
alias of bool

**class** yadm.fields.simple.**FloatField**(*default=<class 'yadm.markers.AttributeNotSet'>*, ***, *choices=None*, ***kwargs*)
Field for float

**type**
alias of float

**class** yadm.fields.simple.**IntegerField**(*default=<class 'yadm.markers.AttributeNotSet'>*, ***, *choices=None*, ***kwargs*)
Field for integer

**type**
alias of int

**class** yadm.fields.simple.**ObjectIdField**(*default_gen=False*)
Field for ObjectId

**Parameters** **default_gen** (*bool*) – generate default value if not set

**type**
alias of ObjectId

**class** yadm.fields.simple.**SimpleField**(*default=<class 'yadm.markers.AttributeNotSet'>*, ***, *choices=None*, ***kwargs*)

Base field for simple types

**Parameters**

- **default** – default value
- **choices** (set) – set of possible values

**class** yadm.fields.simple.**StringField**(*default=<class 'yadm.markers.AttributeNotSet'>*, ***, *choices=None*, ***kwargs*)

Field for string

**type**

alias of `str`

### Datetime field

**class** yadm.fields.datetime.**DatetimeField**(***, *auto_now=False*, ***kwargs*)

Field for time stamp

**Parameters** **auto_now** (*bool*) – datetime.now as default (default: False)

### Decimal field

Field for decimal numbers

This code save to MongoDB document:

**class** yadm.fields.decimal.**DecimalField**(***, *context=None*, ***kwargs*)

Field for work with `decimal.Decimal`

**Parameters**

- **context** (*decimal.Context*) – context for decimal operations (default: run `decimal.getcontext()` when need)
- **default** (*decimal.Decimal*) –

TODO: context in copy()

**context**

Context

**Returns** `decimal.Context` for values

**prepare_value**(*document*, *value*)

Cast value to `decimal.Decimal`

### Embedded documents fields

Work with embedded documents.

```
class EDoc(EmbeddedDocument):
    i = fields.IntegerField()

class Doc(Document):
    __collection__ = 'docs'
    edoc = EmbeddedDocumentField(EDoc)
```

```
doc = Doc()
doc.edoc = EDoc()
doc.edoc.i = 13
db.insert(doc)
```

**class** yadm.fields.embedded.**EmbeddedDocumentField**(*embedded_document_class*, *, *auto_create=True*, ***kwargs*)

Field for embedded objects

**Parameters**

- **embedded_document_class** (EmbeddedDocument) – class for embedded document
- **auto_create** (*bool*) – automatic creation embedded document from access

**copy**()

Return copy of field

### Reference field

Work with references.

```
class RDoc(Document):
    i = fields.IntegerField

class Doc(Document):
    rdoc = fields.ReferenceField(RDoc)

rdoc = RDoc()
rdoc.i = 13
db.insert(rdoc)

doc = Doc()
doc.rdoc = rdoc
db.insert(doc)

doc = db.get_queryset(Doc).with_id(doc.id)  # reload doc
assert doc.rdoc.id == rdoc.id
assert doc.rdoc.i == 13
```

**exception** yadm.fields.reference.**BrokenReference**

Raise if referrenced document is not found

**exception** yadm.fields.reference.**NotBindingToDatabase**

Raise if set ObjectId insted referenced document to new document, who not binded to database.

**class** yadm.fields.reference.**ReferenceField**(*reference_document_class*, ***kwargs*)

Field for work with references

**Parameters** **reference_document_class** – class for refered documents

**get_fake**(*document*, *faker*, *depth*)

Try create referenced document

### Containers fields

Base classes for containers.

**class** yadm.fields.containers.**Container**(*field*, *parent*, *value*)

Base class for containers

**reload**()

Reload all object from database

**class** yadm.fields.containers.**ContainerField**(*item_field=None*, ***, *auto_create=True*, ***kwargs*)

Base class for container fields

**container**

alias of *Container*

**from_mongo**(*document*, *value*)

**get_default**(*document*)

**get_default_value**()

**prepare_item**(*container*, *item*, *value*)

**prepare_value**(*document*, *value*)

**to_mongo**(*document*, *value*)

### List fields

List of objects

```
class Doc(Document):
    __collection__ = 'docs'
    integers = fields.ListField(fields.IntegerField)

doc = Doc()
doc.integers.append(1)
doc.integers.append(2)
assert doc.integers == [1, 2]

db.insert(doc)
doc = db.get_queryset(Doc).with_id(doc.id)  # reload

doc.integers.append(3)  # do not save
assert doc.integers == [1, 2, 3]
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.integers == [1, 2]

doc.integers.remove(2)  # do not save too
assert doc.integers == [1]
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.integers == [1, 2]

doc.integers.push(3)  # $push query
assert doc.integers == [1, 2, 3]
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.integers == [1, 2, 3]

doc.integers.pull(2)  # $pull query
assert doc.integers == [1, 3]
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.integers == [1, 3]
```

**class** yadm.fields.list.**List**(*field*, *parent*, *value*)
Container for list

**append**(*item*)
Append item to list

**Parameters** **item** – item for append

This method does not save object!

**insert**(*index*, *item*)
Append item to list

**Parameters**

- **index** (*int*) –
- **item** – item for insert

This method does not save object!

**pull**(*query*, *reload=True*)
Pull item from database

**Parameters**

- **query** – query for *$pull* on this field
- **reload** (*bool*) – automatically reload all values from database

See *$pull* in MongoDB's *update*.

**push**(*item*, *reload=True*)
Push item directly to database

**Parameters**

- **item** – item for *$push*
- **reload** (*bool*) – automatically reload all values from database

See *$push* in MongoDB's *update*.

**remove**(*item*)
Remove item from list

**Parameters** **item** – item for remove

This method does not save object!

**replace**(*query*, *item*, *reload=True*)
Replace list elements

**Parameters**

- **query** – query for *update*. Keys of this query is relative.
- **item** – embedded document or dict
- **reload** (*bool*) – automatically reload all values from database

**update**(*query*, *values*, *reload=True*)
Update fields in embedded documents

**Parameters**

- **query** – query for *update*. Keys of this query is relative.
- **values** – dict of new values

- **reload** (*bool*) – automatically reload all values from database

**class** yadm.fields.list.**ListField**(*item_field=None*, *, *auto_create=True*, ***kwargs*)
Field for list values

For example, document with list of integers:

```
class TestDoc(Document):
    __collection__ = 'testdoc'
    li = fields.ListField(fields.IntegerField())
```

**container**
alias of *List*

### Set field

Field with sets.

Similar as *yadm.fields.list*.

**class** yadm.fields.set.**Set**(*field*, *parent*, *value*)
Container for set

**add**(*item*)
Append item to set

**Parameters** **item** – item for add

This method does not save object!

**add_to_set**(*item*, *reload=True*)
Add item directly to database

**Parameters**

- **item** – item for *$addToSet*
- **reload** (*bool*) – automatically reload all values from database

See *$addToSet* in MongoDB's *update*.

**discard**(*item*)
Remove item from the set if it is present

**Parameters** **item** – item for discard

This method does not save object!

**pull**(*query*, *reload=True*)
Pull item from database

**Parameters**

- **query** – query for *$pull* on this field
- **reload** (*bool*) – automatically reload all values from database

See *$pull* in MongoDB's *update*.

**remove**(*item*)
Remove item from set

**Parameters** **item** – item for remove

This method does not save object!

**class** yadm.fields.set.**SetField**(*item_field=None*, *, *auto_create=True*, ***kwargs*)
Field for set values

**container**
alias of *Set*

### Map field

Map

```
class Doc(Document):
    __collection__ = 'docs'
    map = fields.MapField(fields.IntegerField)

doc = Doc()
doc.map['a'] = 1
doc.map['b'] = 2
assert doc.map == {'a': 1, 'b': 2}

db.insert(doc)
doc = db.get_queryset(Doc).with_id(doc.id)  # reload

doc.map['c'] = 3  # do not save
assert doc.map == {'a': 1, 'b': 2, 'c': 3}
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.map == {'a': 1, 'b': 2}

del doc.map['b']  # do not save too
assert doc.map == {'a': 1}
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.map == {'a': 1, 'b': 2}

doc.map.set('d', 3)  # $set query
assert doc.map == {'a': 1, 'b': 2, 'c': 3}
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.map == {'a': 1, 'b': 2, 'c': 3}

doc.map.unset('d', 3)  # $unset query
assert doc.map == {'a': 1, 'b': 2}
doc = db.get_queryset(Doc).with_id(doc.id)  # reload
assert doc.map == {'a': 1, 'b': 2}
```

**class** yadm.fields.map.**Map**(*field*, *parent*, *value*)

**set**(*key*, *value*, *reload=True*)
Set key directly in database

**Parameters**

- **key** – key
- **value** – value for *$set*

See *$set* in MongoDB's *set*.

**unset**(*key*, *reload=True*)
Unset key directly in database

**Parameters** **key** – key

See *$unset* in MongoDB's *unset*.

**class** yadm.fields.map.**MapCustomKeysField**(*item_field*, *key_factory*, ***, *key_to_str=<class 'str'>*, *auto_create=True*, ***kwargs*)

Field for maps with custom key type

**Parameters**

- **item_field** (field) –
- **key_factory** (*func*) – function, who return thue key from raw string key
- **key_to_str** (*func*) –
- **auto_create** (*bool*) –

**class** yadm.fields.map.**MapField**(*item_field=None*, ***, *auto_create=True*, ***kwargs*)

Field for maps

**container**

alias of *Map*

### Geo fields

Fields for geo data

See: http://docs.mongodb.org/manual/applications/geospatial-indexes/

GeoJSON: http://geojson.org/geojson-spec.html

**class** yadm.fields.geo.**Geo**

Base class for GeoJSON data

**class** yadm.fields.geo.**GeoCoordinates**

Base class for GeoJSON data with coordinates

**class** yadm.fields.geo.**GeoField**(*types=[<class 'yadm.fields.geo.Point'>, <class 'yadm.fields.geo.MultiPoint'>]*, ***kwargs*)

Base field for GeoJSON objects

**class** yadm.fields.geo.**GeoOneTypeField**(***kwargs*)

Base field for GeoJSON objects with one acceptable type

**class** yadm.fields.geo.**MultiPoint**(*points*)

Class for GeoJSON MultiPoint objects

See: http://geojson.org/geojson-spec.html#id5

**class** yadm.fields.geo.**MultiPointField**(***kwargs*)

Field for MultiPoint

**type**

alias of *MultiPoint*

**class** yadm.fields.geo.**Point**(*longitude*, *latitude*)

Class for GeoJSON Point objects

See: http://geojson.org/geojson-spec.html#id2

**class** yadm.fields.geo.**PointField**(***kwargs*)

Field for Point

**type**

alias of *Point*

## y

yadm.bulk, 11
yadm.database, 5
yadm.documents, 6
yadm.fields, 12
yadm.fields.base, 12
yadm.fields.containers, 16
yadm.fields.datetime, 15
yadm.fields.decimal, 15
yadm.fields.embedded, 15
yadm.fields.geo, 21
yadm.fields.list, 17
yadm.fields.map, 20
yadm.fields.reference, 16
yadm.fields.set, 19
yadm.fields.simple, 14
yadm.join, 12
yadm.queryset, 9
yadm.serialize, 8

## Symbols

__cache__ (yadm.documents.BaseDocument attribute), 7
__changed__ (yadm.documents.BaseDocument attribute), 7
__collection__ (yadm.documents.Document attribute), 7
__data__ (yadm.documents.BaseDocument attribute), 7
__db__ (yadm.documents.Document attribute), 7
__db__ (yadm.documents.DocumentItemMixin attribute), 8
__debug_print__() (yadm.documents.BaseDocument method), 7
__delete__() (yadm.fields.base.FieldDescriptor method), 12
__document__ (yadm.documents.DocumentItemMixin attribute), 8
__fake__() (yadm.documents.BaseDocument method), 7
__field_name__ (yadm.documents.DocumentItemMixin attribute), 8
__get__() (yadm.fields.base.FieldDescriptor method), 12
__get_value__() (yadm.documents.DocumentItemMixin method), 8
__name__ (yadm.documents.DocumentItemMixin attribute), 7
__parent__ (yadm.documents.DocumentItemMixin attribute), 7
__path__ (yadm.documents.DocumentItemMixin attribute), 8
__path_names__ (yadm.documents.DocumentItemMixin attribute), 8
__raw__ (yadm.documents.BaseDocument attribute), 7
__set__() (yadm.fields.base.FieldDescriptor method), 13
__weakref__ (yadm.documents.DocumentItemMixin attribute), 8
_id (yadm.documents.Document attribute), 7

## A

add() (yadm.fields.set.Set method), 19
add_to_set() (yadm.fields.set.Set method), 19
append() (yadm.fields.list.List method), 18

## B

BaseDocument (class in yadm.documents), 7
BooleanField (class in yadm.fields.simple), 14
BrokenReference, 16
Bulk (class in yadm.bulk), 11
bulk() (yadm.database.Database method), 5
bulk() (yadm.queryset.QuerySet method), 9
BulkResult (class in yadm.bulk), 11

## C

Container (class in yadm.fields.containers), 16
container (yadm.fields.containers.ContainerField attribute), 17
container (yadm.fields.list.ListField attribute), 19
container (yadm.fields.map.MapField attribute), 21
container (yadm.fields.set.SetField attribute), 20
ContainerField (class in yadm.fields.containers), 17
context (yadm.fields.decimal.DecimalField attribute), 15
contribute_to_class() (yadm.fields.base.Field method), 13
copy() (yadm.fields.base.Field method), 13
copy() (yadm.fields.embedded.EmbeddedDocumentField method), 16
copy() (yadm.queryset.QuerySet method), 9
count() (yadm.queryset.QuerySet method), 9

## D

Database (class in yadm.database), 5
DatetimeField (class in yadm.fields.datetime), 15
DecimalField (class in yadm.fields.decimal), 15
descriptor_class (yadm.fields.base.Field attribute), 13
discard() (yadm.fields.set.Set method), 19
distinct() (yadm.queryset.QuerySet method), 9
Document (class in yadm.documents), 7
document_class (yadm.fields.base.Field attribute), 13
DocumentItemMixin (class in yadm.documents), 7

## E

EmbeddedDocument (class in yadm.documents), 8
EmbeddedDocumentField (class in yadm.fields.embedded), 16

execute() (yadm.bulk.Bulk method), 11

## F

Field (class in yadm.fields.base), 13
field (yadm.fields.base.FieldDescriptor attribute), 12
FieldDescriptor (class in yadm.fields.base), 12
fields() (yadm.queryset.QuerySet method), 9
fields_all() (yadm.queryset.QuerySet method), 9
find() (yadm.queryset.QuerySet method), 9
find_and_modify() (yadm.queryset.QuerySet method), 10
find_one() (yadm.queryset.QuerySet method), 10
FloatField (class in yadm.fields.simple), 14
from_mongo() (in module yadm.serialize), 8
from_mongo() (yadm.fields.base.Field method), 13
from_mongo() (yadm.fields.containers.ContainerField method), 17

## G

Geo (class in yadm.fields.geo), 21
GeoCoordinates (class in yadm.fields.geo), 21
GeoField (class in yadm.fields.geo), 21
GeoOneTypeField (class in yadm.fields.geo), 21
get_default() (yadm.fields.base.Field method), 13
get_default() (yadm.fields.containers.ContainerField method), 17
get_default_value() (yadm.fields.containers.ContainerField method), 17
get_fake() (yadm.fields.base.Field method), 13
get_fake() (yadm.fields.reference.ReferenceField method), 16
get_if_attribute_not_set() (yadm.fields.base.Field method), 13
get_if_not_loaded() (yadm.fields.base.Field method), 13
get_queryset() (yadm.database.Database method), 6
get_queryset() (yadm.join.Join method), 12

## I

id (yadm.documents.Document attribute), 7
insert() (yadm.bulk.Bulk method), 11
insert() (yadm.database.Database method), 6
insert() (yadm.fields.list.List method), 18
IntegerField (class in yadm.fields.simple), 14

## J

Join (class in yadm.join), 12
join() (yadm.join.Join method), 12
join() (yadm.queryset.QuerySet method), 10

## L

List (class in yadm.fields.list), 17
ListField (class in yadm.fields.list), 19

## M

Map (class in yadm.fields.map), 20
MapCustomKeysField (class in yadm.fields.map), 21
MapField (class in yadm.fields.map), 21
MetaDocument (class in yadm.documents), 7
MultiPoint (class in yadm.fields.geo), 21
MultiPointField (class in yadm.fields.geo), 21

## N

n_inserted (yadm.bulk.BulkResult attribute), 11
n_modified (yadm.bulk.BulkResult attribute), 11
n_removed (yadm.bulk.BulkResult attribute), 11
n_upserted (yadm.bulk.BulkResult attribute), 11
name (yadm.fields.base.Field attribute), 13
name (yadm.fields.base.FieldDescriptor attribute), 12
NotBindingToDatabase, 16
NotLoadedError (class in yadm.fields.base), 12

## O

ObjectIdField (class in yadm.fields.simple), 14

## P

Point (class in yadm.fields.geo), 21
PointField (class in yadm.fields.geo), 21
prepare_item() (yadm.fields.containers.ContainerField method), 17
prepare_value() (yadm.fields.base.Field method), 14
prepare_value() (yadm.fields.containers.ContainerField method), 17
prepare_value() (yadm.fields.decimal.DecimalField method), 15
pull() (yadm.fields.list.List method), 18
pull() (yadm.fields.set.Set method), 19
push() (yadm.fields.list.List method), 18

## Q

QuerySet (class in yadm.queryset), 9

## R

ReferenceField (class in yadm.fields.reference), 16
reload() (yadm.database.Database method), 6
reload() (yadm.fields.containers.Container method), 17
remove() (yadm.database.Database method), 6
remove() (yadm.fields.list.List method), 18
remove() (yadm.fields.set.Set method), 19
remove() (yadm.queryset.QuerySet method), 10
replace() (yadm.fields.list.List method), 18

## S

save() (yadm.database.Database method), 6
Set (class in yadm.fields.set), 19
set() (yadm.fields.map.Map method), 20
SetField (class in yadm.fields.set), 19
SimpleField (class in yadm.fields.simple), 14
sort() (yadm.queryset.QuerySet method), 10

StringField (class in yadm.fields.simple), 15

## T

to_mongo() (in module yadm.serialize), 8
to_mongo() (yadm.fields.base.Field method), 14
to_mongo() (yadm.fields.containers.ContainerField method), 17
type (yadm.fields.geo.MultiPointField attribute), 21
type (yadm.fields.geo.PointField attribute), 21
type (yadm.fields.simple.BooleanField attribute), 14
type (yadm.fields.simple.FloatField attribute), 14
type (yadm.fields.simple.IntegerField attribute), 14
type (yadm.fields.simple.ObjectIdField attribute), 14
type (yadm.fields.simple.StringField attribute), 15

## U

unset() (yadm.fields.map.Map method), 20
update() (yadm.fields.list.List method), 18
update() (yadm.queryset.QuerySet method), 10
update_one() (yadm.database.Database method), 6

## W

with_id() (yadm.queryset.QuerySet method), 11
write_errors (yadm.bulk.BulkResult attribute), 11

## Y

yadm.bulk (module), 11
yadm.database (module), 5
yadm.documents (module), 6
yadm.fields (module), 12
yadm.fields.base (module), 12
yadm.fields.containers (module), 16
yadm.fields.datetime (module), 15
yadm.fields.decimal (module), 15
yadm.fields.embedded (module), 15
yadm.fields.geo (module), 21
yadm.fields.list (module), 17
yadm.fields.map (module), 20
yadm.fields.reference (module), 16
yadm.fields.set (module), 19
yadm.fields.simple (module), 14
yadm.join (module), 12
yadm.queryset (module), 9
yadm.serialize (module), 8