Flashmap server Documentation

Release 1.0

M.C. van den Enk

Feb 23, 2017

CONTENTS

Contents:

CHAPTER

ONE

CONCEPT_MAP MODULE

class concept_map.**ConceptMap**(**args*, ***values*)
Bases: mongoengine.document.Document

A class representing a concept map

Parameters

- **nodes** (*list(Node)*) – a list of nodes (by default all existing node documents)
- **edges** (*list(Edge)*) – a list of edges (by default all existing edge documents)

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception ConceptMap.**MultipleObjectsReturned**
Bases: mongoengine.errors.MultipleObjectsReturned

ConceptMap.**edges**
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: one-to-many-with-listfields

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

ConceptMap.**id**
A field wrapper around MongoDB's ObjectIds.

ConceptMap.**nodes**
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: one-to-many-with-listfields

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

ConceptMap.**objects = [<ConceptMap: ConceptMap object>]**

CHAPTER

TWO

CONSUMER MODULE

class consumer.**Consumer**

Bases: object

This is the class from which the program is controlled. It can be used together with the *handler* module in order to communicate with an external client over a websocket

Parameters

- **concept_map** (ConceptMap) – The concept map object containing references to nodes and edges
- **SOURCES** (*list (str)*) – All of the sources referenced to in the edges of the concept map
- **user** (User) – The active user

add_descriptives(*gender*, *birthdate*, *code*)

Adds the provided descriptives to the active user

Parameters

- **gender** (*str*) – The gender of the user (restricted to 'male', 'female', or 'other')
- **birthdate** (*datetime*) – The date of birth of the user
- **code** (*str*) – A code affirming receiving the informed consent form

add_test(*flashcards_dict*, *items_dict*)

Adds a *test.Test* to the active *user.User*

Parameters

- **flashcards_dict** (*dict (str, str)*) – A dictionary containing flashcard id's and answers
- **items_dict** (*dict (str, str)*) – A dictionary containing test item id's and answers

authenticate(*name*)

A function to either return an existing *user.User* or a new *user.User* based on the given name

Parameters **name** (*str*) – The username

Returns The user with this username

Return type *User*

consumer(*keyword*, *data*)

Pass data to the function corresponding to the provided keyword for the provided user

Parameters

- **keyword** (*str*) – the keyword for which function to use

- **data** (*dict(str, str or dict)*) – the data necessary for executing the function

Returns Contains the keyword and data to send over a websocket to a client

Return type dict(str, str or dict)

create_questionnaire()
Creates a new *questionnaire.Questionnaire*

Returns A new questionnaire

Return type *Questionnaire*

create_test()
Creates a new *test.Test* for the currently active *user.User*

Returns A new test not containing items from previous tests

Return type *Test*

CHAPTER

THREE

EDGE MODULE

class edge.**Edge**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception Edge.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

Edge.**from_node**

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

Edge.**id**

A unicode string field.

Edge.**id_**

A unicode string field.

Edge.**label**
A unicode string field.

Edge.**objects = [<Edge: Edge object>, <Edge: Edge object>, <Edge: Edge object>, <Edge: Edge object>, <Edge: Edg**

Edge.**source**
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: one-to-many-with-listfields

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

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

FOUR

FLASH_INSTANCE MODULE

class flash_instance.**FlashInstance**(**args*, ***kwargs*)

Bases: mongoengine.document.EmbeddedDocument

reference

A reference to *any* Document subclass that will be automatically dereferenced on access (lazily).

Note:

- Any documents used as a generic reference must be registered in the document registry. Importing the model will automatically register it.
- You can use the choices param to limit the acceptable Document types

---

New in version 0.3.

responses

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: one-to-many-with-listfields

---

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

CHAPTER

FIVE

FLASHCARD MODULE

class flashcard.**Flashcard**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception Flashcard.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

Flashcard.**answer**

A unicode string field.

Flashcard.**id**

A field wrapper around MongoDB's ObjectIds.

Flashcard.**objects = []**

Flashcard.**question**

A unicode string field.

Flashcard.**response_model**

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: one-to-many-with-listfields

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

Flashcard.**sources**

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: one-to-many-with-listfields

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

CHAPTER

SIX

FLASHCARD_INSTANCE MODULE

class flashcard_instance.**FlashmapInstance**(**args*, ***kwargs*)

Bases: *flash_instance.FlashInstance*

reference

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

SEVEN

FLASHCARD_RESPONSE MODULE

class flashcard_response.**FlashcardResponse**(**args*, ***kwargs*)

Bases: mongoengine.document.EmbeddedDocument

answer

A unicode string field.

flashcard

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

EIGHT

FLASHEDGE_RESPONSE MODULE

class flashedge_response.**FlashedgeResponse**(**args*, ***kwargs*)

Bases: mongoengine.document.EmbeddedDocument

correct

Boolean field type.

New in version 0.1.2.

end

32-bit integer field.

reference

A reference to *any* Document subclass that will be automatically dereferenced on access (lazily).

Note:

- Any documents used as a generic reference must be registered in the document registry. Importing the model will automatically register it.
- You can use the choices param to limit the acceptable Document types

---

New in version 0.3.

start

32-bit integer field.

CHAPTER

NINE

FLASHMAP_INSTANCE MODULE

class flashmap_instance.**FlashmapInstance**(**args*, ***kwargs*)

Bases: *flash_instance.FlashInstance*

reference

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

TEN

HANDLER MODULE

handler.**handler**(*websocket*, *path*)

Initiate an asyncio thread which receives messages from a client, parse the json file to an object, pass them to consumer() and send the result back to the client

args: websocket – the websocket being used for receiving and sending messages to a client path – the IP address used to host the websocket

CHAPTER

ELEVEN

ITEM_RESPONSE MODULE

class item_response.**ItemResponse**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception ItemResponse.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

ItemResponse.**answer**

A unicode string field.

ItemResponse.**id**

A field wrapper around MongoDB's ObjectIds.

ItemResponse.**item**

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

ItemResponse.**objects** = []

CHAPTER

TWELVE

LOGENTRY MODULE

class logentry.**LogEntry**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception LogEntry.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

LogEntry.**data**

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 {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

LogEntry.**id**

A field wrapper around MongoDB's ObjectIds.

LogEntry.**keyword**

A unicode string field.

LogEntry.**objects** = []

LogEntry.**timestamp**

32-bit integer field.

LogEntry.**user**

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

THIRTEEN

NODE MODULE

class node.**Node**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception Node.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

Node.**id**

A field wrapper around MongoDB's ObjectIds.

Node.**label**

A unicode string field.

Node.**objects = [<Node: Node object>, <Node: Node object>, <Node: Node object>, <Node: Node object>, <Node: No**

CHAPTER

FOURTEEN

QUESTIONNAIRE MODULE

class questionnaire.**Questionnaire**(*questionnaire_items*, ***data*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception Questionnaire.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

Questionnaire.**append_item**(*item*, *answer*)

Questionnaire.**generate_questionniare**()

Questionnaire.**id**

A field wrapper around MongoDB's ObjectIds.

Questionnaire.**objects = []**

Questionnaire.**perceived_ease_of_use_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: one-to-many-with-listfields

---

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

Questionnaire.**perceived_usefulness_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: one-to-many-with-listfields

---

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

CHAPTER

FIFTEEN

QUESTIONNAIRE_ITEM MODULE

class questionnaire_item.**QuestionnaireItem**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception QuestionnaireItem.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

QuestionnaireItem.**id**

A field wrapper around MongoDB's ObjectIds.

QuestionnaireItem.**negative_phrasing**

A unicode string field.

QuestionnaireItem.**objects** = []

QuestionnaireItem.**positive_phrasing**

A unicode string field.

QuestionnaireItem.**usefullness**

Boolean field type.

New in version 0.1.2.

CHAPTER

SIXTEEN

QUESTIONNAIRE_RESPONSE MODULE

class questionnaire_response.**QuestionnaireResponse**(**args*, ***kwargs*)

Bases: mongoengine.document.EmbeddedDocument

answer

32-bit integer field.

questionnaire_item

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

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 Bar(Document):
    content = StringField()
    foo = ReferenceField('Foo')

Foo.register_delete_rule(Bar, 'foo', NULLIFY)
```

Note: *reverse_delete_rule* does not trigger pre / post delete signals to be triggered.

Changed in version 0.5: added *reverse_delete_rule*

CHAPTER

SEVENTEEN

SESSION MODULE

class session.**Session**(**args*, ***kwargs*)

Bases: mongoengine.document.EmbeddedDocument

browser

A unicode string field.

end

32-bit integer field.

end_session()

source_prompted

Boolean field type.

New in version 0.1.2.

start

32-bit integer field.

CHAPTER

EIGHTEEN

TEST MODULE

class test.**Test**(*flashcards*, *items*, *prev_flashcards=[]*, *prev_items=[]*, ***data*)

Bases: mongoengine.document.EmbeddedDocument

append_flashcard(*flashcard*, *answer*)

append_item(*item*, *answer*)

flashcard_responses

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: one-to-many-with-listfields

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

generate_test(*items*, *prev_items*)

item_responses

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: one-to-many-with-listfields

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

CHAPTER

NINETEEN

TEST_ITEM MODULE

class test_item.**TestItem**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception TestItem.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

TestItem.**id**

A field wrapper around MongoDB's ObjectIds.

TestItem.**objects = []**

TestItem.**question**

A unicode string field.

TestItem.**response_model**

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: one-to-many-with-listfields

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

TestItem.**source**

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: one-to-many-with-listfields

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

CHAPTER

TWENTY

USER MODULE

class user.**User**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception User.**MultipleObjectsReturned**

Bases: mongoengine.errors.MultipleObjectsReturned

User.**append_test**(*flashcard_responses*, *item_responses*)

User.**birthdate**

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

User.**code**

A unicode string field.

User.**create_test**(*flashcards*, *items*)

User.**flashmap_condition**

Boolean field type.

New in version 0.1.2.

User.**gender**

A unicode string field.

User.**id**

A field wrapper around MongoDB's ObjectIds.

User.**name**

A unicode string field.

User.**objects**

User.**questionnaire**

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: one-to-many-with-listfields

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

User.**read_sources**
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: one-to-many-with-listfields

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

User.**sessions**
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: one-to-many-with-listfields

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

User.**set_descriptives**(*birthdate*, *gender*, *code*)

User.**tests**
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: one-to-many-with-listfields

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

CHAPTER

TWENTYONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

K

L

N

O

P

Q

R

S

T

U