

---

# **Flashmap server Documentation**

***Release 1.0***

**M.C. van den Enk**

**Feb 23, 2017**



## CONTENTS

1	concept_map module	3
2	consumer module	5
3	edge module	7
4	flash_instance module	9
5	flashcard module	11
6	flashcard_instance module	13
7	flashcard_response module	15
8	flashedge_response module	17
9	flashmap_instance module	19
10	handler module	21
11	item_response module	23
12	logentry module	25
13	node module	27
14	questionnaire module	29
15	questionnaire_item module	31
16	questionnaire_response module	33
17	session module	35
18	test module	37
19	test_item module	39
20	user module	41
21	Indices and tables	43
	Python Module Index	45



Contents:



## 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**>]





## 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*

## 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: Edge object>]

### 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`

## 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 []

---



## 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 []

---





## 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*



## 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*



## 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.



---

## 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*





## 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



## 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 = []
```



## 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*

## 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
```





## 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_questionnaire ()

    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 []

---



## 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.usefulness
        Boolean field type.

        New in version 0.1.2.
```



## 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*



## 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.
```





## 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 []

---



## 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 []

---



## 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 []

---

## INDICES AND TABLES

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





## PYTHON MODULE INDEX

### C

concept\_map, 3  
consumer, 5

### e

edge, 7

### f

flash\_instance, 9  
flashcard, 11  
flashcard\_instance, 13  
flashcard\_response, 15  
flashedge\_response, 17  
flashmap\_instance, 19

### h

handler, 21

### i

item\_response, 23

### l

logentry, 25

### n

node, 27

### q

questionnaire, 29  
questionnaire\_item, 31  
questionnaire\_response, 33

### s

session, 35

### t

test, 37  
test\_item, 39

### u

user, 41

## A

add\_descriptives() (consumer.Consumer method), 5  
 add\_test() (consumer.Consumer method), 5  
 answer (flashcard.Flashcard attribute), 11  
 answer (flashcard\_response.FlashcardResponse attribute), 15  
 answer (item\_response.ItemResponse attribute), 23  
 answer (questionnaire\_response.QuestionnaireResponse attribute), 33  
 append\_flashcard() (test.Test method), 37  
 append\_item() (questionnaire.Questionnaire method), 29  
 append\_item() (test.Test method), 37  
 append\_test() (user.User method), 41  
 authenticate() (consumer.Consumer method), 5

## B

birthdate (user.User attribute), 41  
 browser (session.Session attribute), 35

## C

code (user.User attribute), 41  
 concept\_map (module), 3  
 ConceptMap (class in concept\_map), 3  
 ConceptMap.DoesNotExist, 3  
 ConceptMap.MultipleObjectsReturned, 3  
 Consumer (class in consumer), 5  
 consumer (module), 5  
 consumer() (consumer.Consumer method), 5  
 correct (flashedge\_response.FlashedgeResponse attribute), 17  
 create\_questionnaire() (consumer.Consumer method), 6  
 create\_test() (consumer.Consumer method), 6  
 create\_test() (user.User method), 41

## D

data (logentry.LogEntry attribute), 25

## E

Edge (class in edge), 7  
 edge (module), 7  
 Edge.DoesNotExist, 7  
 Edge.MultipleObjectsReturned, 7

edges (concept\_map.ConceptMap attribute), 3  
 end (flashedge\_response.FlashedgeResponse attribute), 17  
 end (session.Session attribute), 35  
 end\_session() (session.Session method), 35

## F

flash\_instance (module), 9  
 Flashcard (class in flashcard), 11  
 flashcard (flashcard\_response.FlashcardResponse attribute), 15  
 flashcard (module), 11  
 Flashcard.DoesNotExist, 11  
 Flashcard.MultipleObjectsReturned, 11  
 flashcard\_instance (module), 13  
 flashcard\_response (module), 15  
 flashcard\_responses (test.Test attribute), 37  
 FlashcardResponse (class in flashcard\_response), 15  
 flashedge\_response (module), 17  
 FlashedgeResponse (class in flashedge\_response), 17  
 FlashInstance (class in flash\_instance), 9  
 flashmap\_condition (user.User attribute), 41  
 flashmap\_instance (module), 19  
 FlashmapInstance (class in flashcard\_instance), 13  
 FlashmapInstance (class in flashmap\_instance), 19  
 from\_node (edge.Edge attribute), 7

## G

gender (user.User attribute), 41  
 generate\_questionnaire() (questionnaire.Questionnaire method), 29  
 generate\_test() (test.Test method), 37

## H

handler (module), 21  
 handler() (in module handler), 21

## I

id (concept\_map.ConceptMap attribute), 3  
 id (edge.Edge attribute), 7  
 id (flashcard.Flashcard attribute), 11  
 id (item\_response.ItemResponse attribute), 23

id (logentry.LogEntry attribute), 25  
 id (node.Node attribute), 27  
 id (questionnaire.Questionnaire attribute), 29  
 id (questionnaire\_item.QuestionnaireItem attribute), 31  
 id (test\_item.TestItem attribute), 39  
 id (user.User attribute), 41  
 id\_ (edge.Edge attribute), 7  
 item (item\_response.ItemResponse attribute), 23  
 item\_response (module), 23  
 item\_responses (test.Test attribute), 37  
 ItemResponse (class in item\_response), 23  
 ItemResponse.DoesNotExist, 23  
 ItemResponse.MultipleObjectsReturned, 23

## K

keyword (logentry.LogEntry attribute), 25

## L

label (edge.Edge attribute), 7  
 label (node.Node attribute), 27  
 LogEntry (class in logentry), 25  
 logentry (module), 25  
 LogEntry.DoesNotExist, 25  
 LogEntry.MultipleObjectsReturned, 25

## N

name (user.User attribute), 41  
 negative\_phrasing (questionnaire\_item.QuestionnaireItem attribute), 31  
 Node (class in node), 27  
 node (module), 27  
 Node.DoesNotExist, 27  
 Node.MultipleObjectsReturned, 27  
 nodes (concept\_map.ConceptMap attribute), 3

## O

objects (concept\_map.ConceptMap attribute), 3  
 objects (edge.Edge attribute), 8  
 objects (flashcard.Flashcard attribute), 11  
 objects (item\_response.ItemResponse attribute), 23  
 objects (logentry.LogEntry attribute), 25  
 objects (node.Node attribute), 27  
 objects (questionnaire.Questionnaire attribute), 29  
 objects (questionnaire\_item.QuestionnaireItem attribute), 31  
 objects (test\_item.TestItem attribute), 39  
 objects (user.User attribute), 41

## P

perceived\_ease\_of\_use\_items (questionnaire.Questionnaire attribute), 29

perceived\_usefulness\_items (questionnaire.Questionnaire attribute), 29  
 positive\_phrasing (questionnaire\_item.QuestionnaireItem attribute), 31

## Q

question (flashcard.Flashcard attribute), 11  
 question (test\_item.TestItem attribute), 39  
 Questionnaire (class in questionnaire), 29  
 questionnaire (module), 29  
 questionnaire (user.User attribute), 41  
 Questionnaire.DoesNotExist, 29  
 Questionnaire.MultipleObjectsReturned, 29  
 questionnaire\_item (module), 31  
 questionnaire\_item (questionnaire\_response.QuestionnaireResponse attribute), 33  
 questionnaire\_response (module), 33  
 QuestionnaireItem (class in questionnaire\_item), 31  
 QuestionnaireItem.DoesNotExist, 31  
 QuestionnaireItem.MultipleObjectsReturned, 31  
 QuestionnaireResponse (class in questionnaire\_response), 33

## R

read\_sources (user.User attribute), 42  
 reference (flash\_instance.FlashInstance attribute), 9  
 reference (flashcard\_instance.FlashmapInstance attribute), 13  
 reference (flashedge\_response.FlashedgeResponse attribute), 17  
 reference (flashmap\_instance.FlashmapInstance attribute), 19  
 response\_model (flashcard.Flashcard attribute), 11  
 response\_model (test\_item.TestItem attribute), 39  
 responses (flash\_instance.FlashInstance attribute), 9

## S

Session (class in session), 35  
 session (module), 35  
 sessions (user.User attribute), 42  
 set\_descriptives() (user.User method), 42  
 source (edge.Edge attribute), 8  
 source (test\_item.TestItem attribute), 39  
 source\_prompted (session.Session attribute), 35  
 sources (flashcard.Flashcard attribute), 11  
 start (flashedge\_response.FlashedgeResponse attribute), 17  
 start (session.Session attribute), 35

## T

Test (class in test), 37  
 test (module), 37

test\_item (module), [39](#)  
TestItem (class in test\_item), [39](#)  
TestItem.DoesNotExist, [39](#)  
TestItem.MultipleObjectsReturned, [39](#)  
tests (user.User attribute), [42](#)  
timestamp (logentry.LogEntry attribute), [25](#)  
to\_node (edge.Edge attribute), [8](#)

## U

usefulness (questionnaire\_item.QuestionnaireItem attribute), [31](#)  
User (class in user), [41](#)  
user (logentry.LogEntry attribute), [25](#)  
user (module), [41](#)  
User.DoesNotExist, [41](#)  
User.MultipleObjectsReturned, [41](#)