
mogwai Documentation

Release 0.7.5

Cody Lee

August 07, 2015

1	Features	3
2	Links	5
3	Download	7
4	Contents	9
4.1	Quick Start	9
4.2	Reference	11
4.3	Examples	57
4.4	ChangeLog	69
4.5	Contributing	73
4.6	Ideas	73
5	Indices and tables	75
	Python Module Index	77

mogwai is an object graph mapper (OGM) designed for Titan Graph Database.

Features

- Straightforward syntax
- Query Support
- Vertex and Edge Support
- Uses RexPro Connection Pooling
- Gevent and Eventlet socket support via RexPro
- Supports Relationship quick-modeling
- Supports custom gremlin scripts and methods
- Doesn't restrict the developer from using direct methods.
- Save Strategies
- Property Validation
- Optionally can utilize [factory_boy](#) to generate models.
- Interactive shell available
- Performance monitoring tools available.
- Serialization support for Pickle
- Support for un-modeled properties with dictionary-like access
- Support for imports via groovy and the groovy Script Engine. (*See Example*)
- Tested with Python 2.7, 3.3 and 3.4

Links

- Documentation: <http://mogwai.readthedocs.org/>
- Official Repository: <https://bitbucket.org/wellaware/mogwai.git>
- Package: <https://pypi.python.org/pypi/mogwai/>

mogwai is known to support Python 2.7, 3.3 and 3.4.

Download

PyPI: <https://pypi.python.org/pypi/mogwai/>

```
$ pip install mogwai
```

Source: <https://bitbucket.org/wellaware/mogwai.git>

```
$ git clone git://bitbucket.org/wellaware/mogwai.git
$ python setup.py install
```

4.1 Quick Start

4.1.1 Usage

Note: This section provides a quick summary of mogwai features. A more detailed listing is available in the full documentation.

4.1.2 Setup Connection

You'll need to setup the connection to the graph database. Mogwai handles connection pooling for you, but you should handle load balancing using dedicated equipment.

```
from mogwai.connection import setup

# Without Authentication
setup('localhost')
# With authentication
#setup('localhost', username='rexster', password='rexster')
# With gevent support
#setup('localhost', concurrency='gevent') # default is Standard Synchronous Python Sockets
# With eventlet support
#setup('localhost', concurrency='eventlet') # default is Standard Synchronous Python Sockets
```

4.1.3 Defining Models

Models declare a Vertex or Edge model and attributes it should possibly contain.

```
from mogwai.models import Vertex, Edge
from mogwai.properties import String, Integer

_val = 0

def counter():
    global _val
    _val += 1
    return _val
```

```
class TestVertexModel(Vertex):
    element_type = 'test_vertex_model'

    name = String(default='test_vertex')
    test_val = Integer(default=counter)

class TestEdgeModel(Edge):
    label = 'test_edge_model'

    name = String(default='test_edge')
    test_val = Integer(default=counter)
```

4.1.4 Using Models

mogwai models can be utilized in a similar fashion to the way the Django ORM was constructed, but tailored for Graph Databases.

Creating Vertex or Edge

```
my_vertex_1 = TestVertexModel.create(name='my vertex')
my_vertex_2 = TestVertexModel.create(name='my other vertex')
my_edge = TestEdgeModel.create(outV=my_vertex_1, inV=my_vertex_2, name='my edge')
```

Retrieving a Vertex or Edge

```
# Get all the TestVertexModel Vertices
vertices = TestVertexModel.all()
# Get a subset of vertices by titan ID
vertices = TestVertexModel.all([1234, 5678, 9012])
# Get a vertex by titan ID
vertex = TestVertexModel.get(1234)

# Getting all Edges isn't currently supported
# Get a subset of edges by titan IDs
edges = TestEdgeModel.all(['123-UX4', '215-PX3', '95U-32Z'])
# get a single edge by titan ID
edge = TestEdgeModel.get('123-UX4')

# Get edge between two vertices
edge = TestEdgeModel.get_between(outV=my_vertex_1, inV=my_vertex_2)
```

Simple Traversals

Vertex Traversals

```
# Get All Edges from the vertex
edges = my_vertex_1.bothE()
# Get outgoing edges from the vertex
edges = my_vertex_1.outE()
# Get incoming edges to the vertex
```

```

edges = my_Vertex_1.inE()
# Specify an edge type for any edge traversal operation (works for outE, inE, bothE)
## By using models
test_model_edges = my_vertex_1.outE(TestEdgeModel)
## or by using manual labels
test_model_edges = my_vertex_1.outE('test_edge_model')

# Get all Vertices connected to the vertex
vertices = my_vertex_1.bothV()
# Get all vertices who are connected by edge coming into the current vertex (note uni-directed edges)
vertices = my_vertex_1.outV()
# Get all vertices who are connected by edge coming from the current vertex (note uni-directed edges)
vertices = my_vertex_1.inV()
# Specify an edge type for any edge traversal operation (works for outV, inV, bothV)
## By using models
test_model_vertices = my_vertex_1.outV(TestEdgeModel)
## or by using manual element types
test_model_vertices = my_vertex_1.outV('test_edge_model')

```

Edge Traversals

```

# Get the vertex which is from the outgoing side of the edge
vertex = my_edge.outV()
# Get the vertex which is from the incoming side of the edge
vertex = my_edge.inV()

```

4.2 Reference

4.2.1 Connection

`mogwai.connection.execute_query(*args, **kwargs)`

Execute a raw Gremlin query with the given parameters passed in.

Parameters

- **query** (*str*) – The Gremlin query to be executed
- **params** (*dict*) – Parameters to the Gremlin query
- **connection** (*RexPro(Sync|Gevent|Eventlet)Connection or None*) – The RexPro connection to execute the query with
- **context** – String context data to include with the query for stats logging

Return type dict

`mogwai.connection.generate_spec()`

Generates a titan index and type specification document based on loaded Vertex and Edge models

`mogwai.connection.get_response(query, params, isolate, transaction, connection, connection_pool)`

`mogwai.connection.pop_execute_query_kwargs(keyword_arguments)`

pop the optional execute query arguments from arbitrary kwargs; return non-None query kwargs in a dict

`mogwai.connection.setup` (*host*, *graph_name*=u'graph', *graph_obj_name*=u'g', *username*=u'', *password*=u'', *metric_reporters*=None, *pool_size*=10, *concurrency*=u'sync')

Sets up the connection, and instantiates the models

`mogwai.connection.sync_spec` (*filename*, *host*, *graph_name*=u'graph', *graph_obj_name*=u'g', *username*=u'', *password*=u'', *dry_run*=False)

Sync the given spec file to mogwai.

Parameters

- **filename** (*str*) – The filename of the spec file
- **host** (*str*) – The host the be synced
- **graph_name** (*str*) – The name of the graph to be synced
- **dry_run** (*boolean*) – Only prints generated Gremlin if True

Returns None

4.2.2 Vertex

class `mogwai.models.vertex.EnumVertexBaseMeta`

This metaclass allows you to access MyVertexModel as if it were an enum. Ex. MyVertexModel.FOO

The values are cached in a dictionary. This is useful if the number of MyVertexModels is small, however it it grows too large, you should be doing it a different way.

This looks for a special (optional) function named *enum_generator* in your model and calls that to generate the ENUM for the model.

There is an additional optional model attribute that can be set *__enum_id_only__* (defaults to True) which dictates whether or not just the Vertex ID is stored, or the whole Vertex in cache.

enums = None

mro () → list

return a type's method resolution order

class `mogwai.models.vertex.Vertex` (***values*)

The Vertex model base class.

The element type is auto-generated from the subclass name, but can optionally be set manually

exception `DoesNotExist`

Object not found in database

args

message

`Vertex.FACTORY_CLASS = None`

exception `Vertex.MultipleObjectsReturned`

Multiple objects returned on unique key lookup

args

message

exception `Vertex.WrongElementType`

Unique lookup with key corresponding to vertex of different type

args

message

classmethod `Vertex.all` (*ids=[]*, *as_dict=False*, *match_length=True*, *args, **kwargs)

Load all vertices with the given ids from the graph. By default this will return a list of vertices but if `as_dict` is `True` then it will return a dictionary containing ids as keys and vertices found as values.

Parameters

- **ids** (*list*) – A list of titan ids
- **as_dict** (*boolean*) – Toggle whether to return a dictionary or list

Return type dict | list

`Vertex.as_dict` ()

Returns a map of column names to cleaned values

Return type dict

`Vertex.as_save_params` ()

Returns a map of property names to cleaned values containing only the properties which should be persisted on save.

Return type dict

`Vertex.bothE` (*labels, **kwargs)

Return a list of edges both incoming and outgoing from this vertex.

Parameters

- **label** (*str or BaseEdge or None*) – The edge label to be traversed (optional)
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return
- **types** (*list*) – A list of allowed element types

`Vertex.bothV` (*labels, **kwargs)

Return a list of vertices both incoming and outgoing from this vertex.

Parameters

- **label** (*str or BaseEdge or None*) – The edge label to be traversed (optional)
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return
- **types** (*list*) – A list of allowed element types

`Vertex.create` (*args, **kwargs)

Create a new element with the given information.

`Vertex.delete` ()

Delete the current vertex from the graph.

`Vertex.delete_inE` (*labels)

Delete all incoming edges with the given label.

`Vertex.delete_inV` (*labels)

Delete all incoming vertices connected with edges with the given label.

`Vertex.delete_outE` (*labels)

Delete all outgoing edges with the given label.

`Vertex.delete_outV (*labels)`

Delete all outgoing vertices connected with edges with the given label.

`Vertex.deserialize (data)`

Deserializes rexpro response into vertex or edge objects

`Vertex.element_type = None`

classmethod `Vertex.find_by_value (field, value, as_dict=False)`

Returns vertices that match the given field/value pair.

Parameters

- **field** (*str*) – The field to search
- **value** (*str*) – The value of the field
- **as_dict** (*boolean*) – Return results as a dictionary

Return type [mogwai.models.Vertex]

classmethod `Vertex.get (id, *args, **kwargs)`

Look up vertex by its ID. Raises a DoesNotExist exception if a vertex with the given vid was not found. Raises a MultipleObjectsReturned exception if the vid corresponds to more than one vertex in the graph.

Parameters **id** (*str*) – The ID of the vertex

Return type mogwai.models.Vertex

classmethod `Vertex.get_element_type ()`

Returns the element type for this vertex.

@returns: str

`Vertex.get_property_by_name (key)`

Get's the db_field_name of a property by key

Parameters **key** (*basestring | str*) – attribute of the model

Return type basestring | str | None

`Vertex.gremlin_path = u'vertex.groovy'`

`Vertex.id`

`Vertex.inE (*labels, **kwargs)`

Return a list of edges with the given label coming into this vertex.

Parameters

- **label** (*str or BaseEdge*) – The edge label to be traversed
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return
- **types** (*list*) – A list of allowed element types

`Vertex.inV (*labels, **kwargs)`

Return a list of vertices reached by traversing the incoming edge with the given label.

Parameters

- **label** (*str or BaseEdge*) – The edge label to be traversed
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return

- **types** (*list*) – A list of allowed element types

Vertex.**items** ()

Vertex.**keys** ()

Vertex.**outE** (*labels, **kwargs)

Return a list of edges with the given label going out of this vertex.

Parameters

- **label** (*str or BaseEdge*) – The edge label to be traversed
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return
- **types** (*list*) – A list of allowed element types

Vertex.**outV** (*labels, **kwargs)

Return a list of vertices reached by traversing the outgoing edge with the given label.

Parameters

- **labels** (*str or BaseEdge*) – pass in the labels to follow in as positional arguments
- **limit** (*int or None*) – The number of the page to start returning results at
- **offset** (*int or None*) – The maximum number of results to return
- **types** (*list*) – A list of allowed element types

Vertex.**pre_save** ()

Pre-save hook which is run before saving an element

Vertex.**pre_update** (**values)

Override this to perform pre-update validation

Vertex.**query** ()

Vertex.**reload** (*args, **kwargs)

Reload the given element from the database.

Vertex.**save** (*args, **kwargs)

Save the current vertex using the configured save strategy, the default save strategy is to re-save all fields every time the object is saved.

Vertex.**translate_db_fields** (*data*)

Translates field names from the database into field names used in our model

this is for cases where we're saving a field under a different name than it's model property

Parameters *data* – dict

Return type dict

Vertex.**update** (**values)

performs an update of this element with the given values and returns the saved object

Vertex.**validate** ()

Cleans and validates the field values

Vertex.**validate_field** (*field_name, val*)

Perform the validations associated with the field with the given name on the value passed.

Parameters

- **field_name** (*str*) – The name of property whose validations will be run

- **val** (*mixed*) – The value to be validated

Vertex.**values** ()

class mogwai.models.vertex.**VertexMetaClass**
Metaclass for vertices.

mro () → list
return a type's method resolution order

4.2.3 Edge

class mogwai.models.edge.**Edge** (*outV, inV, **values*)
Base class for all edges.

exception DoesNotExist
Object not found in database

args

message

Edge.**FACTORY_CLASS** = None

exception Edge.**MultipleObjectsReturned**
Multiple objects returned on unique key lookup

args

message

exception Edge.**WrongElementType**
Unique lookup with key corresponding to vertex of different type

args

message

classmethod Edge.**all** (*ids, as_dict=False, *args, **kwargs*)
Load all edges with the given edge_ids from the graph. By default this will return a list of edges but if as_dict is True then it will return a dictionary containing edge_ids as keys and edges found as values.

Parameters

- **ids** (*list*) – A list of titan IDs
- **as_dict** (*boolean*) – Toggle whether to return a dictionary or list

Return type dict | list

Edge.**as_dict** ()
Returns a map of column names to cleaned values

Return type dict

Edge.**as_save_params** ()
Returns a map of property names to cleaned values containing only the properties which should be persisted on save.

Return type dict

classmethod Edge.**create** (*outV, inV, label=None, *args, **kwargs*)
Create a new edge of the current type coming out of vertex outV and going into vertex inV with the given properties.

Parameters

- **outV** (*Vertex*) – The vertex the edge is coming out of
- **inV** (*Vertex*) – The vertex the edge is going into

Edge.**delete** ()

Delete the current edge from the graph.

Edge.**deserialize** (*data*)

Deserializes rexpri response into vertex or edge objects

classmethod Edge.**find_by_value** (*field, value, as_dict=False*)

Returns edges that match the given field/value pair.

Parameters

- **field** (*str*) – The field to search
- **value** (*str*) – The value of the field
- **as_dict** (*boolean*) – Return results as a dictionary

Return type [mogwai.models.Edge]

classmethod Edge.**get** (*id, *args, **kwargs*)

Look up edge by titan assigned ID. Raises a DoesNotExist exception if a edge with the given edge id was not found. Raises a MultipleObjectsReturned exception if the edge_id corresponds to more than one edge in the graph.

Parameters **id** (*str | basestring*) – The titan assigned ID

Return type mogwai.models.Edge

classmethod Edge.**get_between** (*outV, inV, page_num=None, per_page=None*)

Return all the edges with a given label between two vertices.

Parameters

- **outV** (*Vertex*) – The vertex the edge comes out of.
- **inV** (*Vertex*) – The vertex the edge goes into.
- **page_num** (*int*) – The page number of the results
- **per_page** (*int*) – The number of results per page

Return type list

classmethod Edge.**get_label** ()

Returns the label for this edge.

Return type str

Edge.**get_property_by_name** (*key*)

Get's the db_field_name of a property by key

Parameters **key** (*basestring | str*) – attribute of the model

Return type basestring | str | None

Edge.**gremlin_path** = 'edge.groovy'

Edge.**id**

Edge.**inV** (**args, **kwargs*)

Return the vertex that this edge goes into.

Return type *Vertex*

Edge.**items** ()

Edge.**keys** ()

Edge.**label** = None

Edge.**outV** (*args, **kwargs)

Return the vertex that this edge is coming out of.

Return type *Vertex*

Edge.**pre_save** ()

Pre-save hook which is run before saving an element

Edge.**pre_update** (**values)

Override this to perform pre-update validation

Edge.**reload** (*args, **kwargs)

Reload the given element from the database.

Edge.**save** (*args, **kwargs)

Save this edge to the graph database.

Edge.**translate_db_fields** (data)

Translates field names from the database into field names used in our model

this is for cases where we're saving a field under a different name than it's model property

Parameters *data* – dict

Return type dict

Edge.**update** (**values)

performs an update of this element with the given values and returns the saved object

Edge.**validate** ()

Perform validation of this edge raising a ValidationError if any problems are encountered.

Edge.**validate_field** (field_name, val)

Perform the validations associated with the field with the given name on the value passed.

Parameters

- **field_name** (*str*) – The name of property whose validations will be run
- **val** (*mixed*) – The value to be validated

Edge.**values** ()

class mogwai.models.edge.**EdgeMetaClass**

Metaclass for edges.

mro () → list

return a type's method resolution order

4.2.4 Gremlin

class mogwai.gremlin.base.**BaseGremlinMethod** (path=None, method_name=None, class-method=False, property=False, defaults=None, transaction=True, imports=None)

Maps a function in a groovy file to a method on a python class

configure_method (*klass*, *attr_name*, *gremlin_path*)

Sets up the methods internals

Parameters

- **klass** (*object*) – The class object this function is being added to
- **attr_name** (*str*) – The attribute name this function will be added as
- **gremlin_path** (*str*) – The path to the gremlin file containing method

transform_params_to_database (*params*)

Takes a dictionary of parameters and recursively translates them into parameters appropriate for sending over REXPRO.

Parameters **params** (*dict*) – The parameters to be sent to the function

Return type dict

class mogwai.gremlin.base.**GremlinMethod** (*path=None*, *method_name=None*, *classmethod=False*, *property=False*, *defaults=None*, *transaction=True*, *imports=None*)

Gremlin method that returns a graph element

configure_method (*klass*, *attr_name*, *gremlin_path*)

Sets up the methods internals

Parameters

- **klass** (*object*) – The class object this function is being added to
- **attr_name** (*str*) – The attribute name this function will be added as
- **gremlin_path** (*str*) – The path to the gremlin file containing method

transform_params_to_database (*params*)

Takes a dictionary of parameters and recursively translates them into parameters appropriate for sending over REXPRO.

Parameters **params** (*dict*) – The parameters to be sent to the function

Return type dict

class mogwai.gremlin.base.**GremlinTable** (*path=None*, *method_name=None*, *classmethod=False*, *property=False*, *defaults=None*, *transaction=True*, *imports=None*)

Gremlin method that returns a table as its result

configure_method (*klass*, *attr_name*, *gremlin_path*)

Sets up the methods internals

Parameters

- **klass** (*object*) – The class object this function is being added to
- **attr_name** (*str*) – The attribute name this function will be added as
- **gremlin_path** (*str*) – The path to the gremlin file containing method

transform_params_to_database (*params*)

Takes a dictionary of parameters and recursively translates them into parameters appropriate for sending over REXPRO.

Parameters **params** (*dict*) – The parameters to be sent to the function

Return type dict

class mogwai.gremlin.base.**GremlinValue** (*path=None, method_name=None, classmethod=False, property=False, defaults=None, transaction=True, imports=None*)

Gremlin Method that returns one value

configure_method (*klass, attr_name, gremlin_path*)

Sets up the methods internals

Parameters

- **klass** (*object*) – The class object this function is being added to
- **attr_name** (*str*) – The attribute name this function will be added as
- **gremlin_path** (*str*) – The path to the gremlin file containing method

transform_params_to_database (*params*)

Takes a dictionary of parameters and recursively translates them into parameters appropriate for sending over REXPRO.

Parameters **params** (*dict*) – The parameters to be sent to the function

Return type dict

mogwai.gremlin.groovy.**GroovyFileDef**

alias of GroovyFileDefinition

class mogwai.gremlin.groovy.**GroovyFunction** (*name, args, body, defn*)

args

Alias for field number 1

body

Alias for field number 2

count (*value*) → integer – return number of occurrences of value

defn

Alias for field number 3

index (*value* [, *start* [, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

name

Alias for field number 0

class mogwai.gremlin.groovy.**GroovyFunctionParser**

Given a string containing a single function definition this class will parse the function definition and return information regarding it.

FuncDefn = {{{{{{“def” Re:('[A-Za-z_]\w*')} “{” Re:('[A-Za-z_]\w*') [, Re:('[A-Za-z_]\w*')...} ”} } “{”

FuncName = Re:('[A-Za-z_]\w*')

KeywordDef = “def”

VarName = Re:('[A-Za-z_]\w*')

classmethod **parse** (*data*)

Parse the given function definition and return information regarding the contained definition.

Parameters **data** (*str | basestring*) – The function definition in a string

Return type dict

class `mogwai.gremlin.groovy.GroovyImport` (*comment_list*, *import_strings*, *import_list*)

comment_list

Alias for field number 0

count (*value*) → integer – return number of occurrences of value

import_list

Alias for field number 2

import_strings

Alias for field number 1

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

class `mogwai.gremlin.groovy.GroovyImportParser`

Given a string containing a single import definition this class will parse the import definition and return information regarding it.

CommentVar = `comment`

ImportDef = `Suppress("import")`

ImportDefn = `{{{Suppress("import") Re:("[A-Za-z_*]*") [. Re:("[A-Za-z_*]*")]}...} Suppress(";")} [{{Suppress("//"`

ImportVarName = `Re:("[A-Za-z_*]*")`

OptionalSpace = `[" "]`

classmethod `parse` (*data*)

Parse the given import and return information regarding the contained import statement.

Parameters *data* (*str* | *basestring*) – The import statement in a string

Return type dict

`mogwai.gremlin.groovy.parse` (*filename*)

Parse Groovy code in the given file and return a list of information about each function necessary for usage in queries to database.

Parameters *filename* (*str*) – The file containing groovy code.

Return type list

class `mogwai.gremlin.table.Table` (*gremlin_result*)

A table accepts the results of a GremlinTable in it's constructor. It can be iterated over like a normal list, but within the rows the dictionaries are accessible via `.notation`

For example:

```
# returns a table of people & my friend edge to them # the edge contains my nickname for that person
friends = mogwai.gremlin.GremlinTable()
```

```
def get_friends_and_my_nickname(self): result = self.friends() for i in result:
```

```
    print "{:}" .format(i.friend_edge.nickname, i.person.name)
```

```
next ()
```

class `mogwai.gremlin.table.Row` (*data*)

A row represent a table row, from which it's columns can be accessed like a tuple or dict. Rows are read-only and accept elements or dicts with as initializers as a result of a GremlinTable query. Also the `.getattr` notation can be used to access elements

```
Example: row = Row({'person': Friend.create(...), 'myval': 3}) print "{:} - {}".format(row.friend_edge.nickname, row.person.name, row.myval)
```

```
items ()
```

```
iteritems ()
```

```
keys ()
```

```
next ()
```

```
values ()
```

4.2.5 Properties

```
class mogwai.properties.base.BaseValueManager (graph_property, value, strategy=<class 'mogwai.properties.strategy.SaveAlways'>)
```

Value managers are used to manage values pulled from the database and track state changes.

These are useful for save strategies.

```
changed
```

Indicates whether or not this value has changed.

Return type bool

```
deleted
```

Indicates whether or not this value has been deleted.

Return type bool

```
delval ()
```

Delete a given value

```
get_property ()
```

Returns a value-managed property attributes

Return type property

```
getval ()
```

Return the current value.

```
previous_value
```

```
setval (val)
```

Updates the current value.

param val The new value

```
class mogwai.properties.base.GraphProperty (description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u'')
```

Base class for graph property types

```
can_delete
```

```
data_type = u'Object'
```

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

classmethod get_value_from_choices (value, choices)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0**set_db_field_prefix (prefix, override=False)**

Sets the graph property name prefix during document class construction.

set_property_name (name)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (first_save=False)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (value)

Converts python value into database value

Return type Object

to_python (value)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (value)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.BaseValidator object>

value_manageralias of *BaseValueManager*

```
class mogwai.properties.properties.Boolean (description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u')
```

Boolean Data property type

can_delete**data_type = 'Boolean'****db_field_name**

Returns the name of the mogwai name of this graph property

Return type basestring | str**get_default ()**

Returns the default value for this graph property if one is available.

Return type Object | None**get_save_strategy ()**

Returns the save strategy attached to this graph property.

Return type Callable**get_value_from_choices (value, choices)**

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.**Parameters** **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict**Return type** Object**has_db_field_prefix**

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool**instance_counter = 0****set_db_field_prefix (prefix, override=False)**

Sets the graph property name prefix during document class construction.

set_property_name (name)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`**Parameters** **name** (*str*) – The name of this graph property**should_save (first_save=False)**

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool**to_database (value)**

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type `Object`

validator = `<mogwai.properties.validators.BooleanValidator object>`

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.DateTime` (*strict=True, **kwargs*)

UTC DateTime Data property type

can_delete

data_type = `'Double'`

db_field_name

Returns the name of the mogwai name of this graph property

Return type `basestring | str`

get_default ()

Returns the default value for this graph property if one is available.

Return type `Object | None`

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type `Callable`

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type `Object`

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type `bool`

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.DateTimeUTCValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.DateTimeNaive` (*strict=True, **kwargs*)

DateTime Data property type

can_delete

data_type = 'Double'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a ValidationError if there's a problem

Return type Object

validator = <mogwai.properties.validators.DateTimeValidator object>

value_manager

alias of BaseValueManager

class mogwai.properties.properties.**Decimal** (*description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u''*)

Decimal Data property type

can_delete

data_type = u'Object'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type `bool`

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type `Object`

validator = `<mogwai.properties.validators.DecimalValidator object>`

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.Dictionary` (*description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u''*)

Dictionary Data property type

can_delete

data_type = `'HashMap'`

db_field_name

Returns the name of the mogwai name of this graph property

Return type `basestring | str`

get_default ()

Returns the default value for this graph property if one is available.

Return type `Object | None`

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type `Callable`

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type `Object`

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0**set_db_field_prefix** (*prefix*, *override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

Converts python value into database value

Return type Object

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.DictValidator object>**value_manager**

alias of `BaseValueManager`

class `mogwai.properties.properties.Double` (***kwargs*)

Double Data property type

can_delete**data_type = 'Double'****db_field_name**

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.FloatValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.Email` (**args, **kwargs*)

Email Data property type

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

Converts python value into database value

Return type Object

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (*value*)

validator = <mogwai.properties.validators.EmailValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.Float` (**kwargs)

Float class for backwards compatability / if you really want to

can_delete

data_type = 'Double'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.FloatValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.IPv4` (**args, **kwargs*)

IPv4 Data property type

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (value, choices)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0**set_db_field_prefix (prefix, override=False)**

Sets the graph property name prefix during document class construction.

set_property_name (name)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (first_save=False)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (value)

Converts python value into database value

Return type Object

to_python (value)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (value)**validator = <mogwai.properties.validators.RegexValidator object>****value_manager**

alias of `BaseValueManager`

class `mogwai.properties.properties.IPv6` (**args, **kwargs*)
IPv6 Data property type

can_delete

data_type = 'String'

db_field_name
Returns the name of the mogwai name of this graph property
Return type basestring | str

get_default ()
Returns the default value for this graph property if one is available.
Return type Object | None

get_save_strategy ()
Returns the save strategy attached to this graph property.
Return type Callable

get_value_from_choices (*value, choices*)
Returns the key for the choices tuple of tuples
Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.
Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict
Return type Object

has_db_field_prefix
Determines if a field prefix has already been defined.

has_default
Indicates whether or not this graph property has a default value.
Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)
Sets the graph property name prefix during document class construction.

set_property_name (*name*)
Sets the graph property name during document class construction.
This value will be ignored if `db_field` is set in `__init__`
Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)
Indicates whether or not the property should be saved based on it's save strategy.
Return type bool

to_database (*value*)
Converts python value into database value
Return type Object

to_python (*value*)
Converts data from the database into python values raises a `ValidationError` if the value can't be converted
Return type Object

validate (*value*)

validator = <mogwai.properties.validators.RegexValidator object>

value_manager

alias of BaseValueManager

class mogwai.properties.properties.**IPV6WithV4** (**args, **kwargs*)

IPv6 with Mapped/Translated/Embedded IPv4 Data property type

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

Converts python value into database value

Return type Object

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (*value*)

validator = <mogwai.properties.validators.RegexValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.Integer` (*description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u''*)

Integer Data property type

can_delete

data_type = 'Integer'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.LongValidator object>

value_manager

alias of `BaseValueManager`

```
class mogwai.properties.properties.List (description=None, primary_key=False, index=False,
index_ext=None, db_field=None, choices=None,
default=None, required=False, save_strategy=<class
'mogwai.properties.strategy.SaveAlways'>,
unique=None, db_field_prefix=u')
```

List Data property type

can_delete

data_type = 'ArrayList'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if db_field is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

Converts python value into database value

Return type Object

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.ListValidator object>

value_manager

alias of `BaseValueManager`

class mogwai.properties.properties.**Long** (*description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u''*)

Long Data property type

can_delete

data_type = 'Long'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.LongValidator object>

value_manager

alias of `BaseValueManager`

```
class mogwai.properties.properties.PositiveInteger (description=None,           pri-
                                                    mary_key=False,           in-
                                                    dex=False,           index_ext=None,
                                                    db_field=None,           choices=None,
                                                    default=None,           required=False,
                                                    save_strategy=<class      'mog-
wai.properties.strategy.SaveAlways'>,
                                                    unique=None, db_field_prefix=u'')
```

Positive Integer Data property type

can_delete

data_type = 'Integer'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.PositiveIntegerValidator object>

value_manager

alias of `BaseValueManager`

```
class mogwai.properties.properties.PositiveLong (description=None, primary_key=False,
index=False, index_ext=None,
db_field=None, choices=None,
default=None, required=False,
save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>,
unique=None, db_field_prefix=u'')
```

Positive Long Data property type

can_delete

data_type = 'Long'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (value, choices)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (prefix, override=False)

Sets the graph property name prefix during document class construction.

set_property_name (name)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (first_save=False)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (value)

to_python (value)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type `Object`

validator = `<mogwai.properties.validators.PositiveIntegerValidator object>`

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.Short` (*description=None, primary_key=False, index=False, index_ext=None, db_field=None, choices=None, default=None, required=False, save_strategy=<class 'mogwai.properties.strategy.SaveAlways'>, unique=None, db_field_prefix=u''*)

Short Data property type

can_delete

data_type = `'Short'`

db_field_name

Returns the name of the mogwai name of this graph property

Return type `basestring | str`

get_default ()

Returns the default value for this graph property if one is available.

Return type `Object | None`

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type `Callable`

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters *value* (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type `Object`

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type `bool`

instance_counter = `0`

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters *name* (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a ValidationError if there's a problem

Return type Object

validator = <mogwai.properties.validators.IntegerValidator object>

value_manager

alias of BaseValueManager

class mogwai.properties.properties.**Slug** (**args, **kwargs*)

Slug Data property type

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters `name` (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type `bool`

to_database (*value*)

Converts python value into database value

Return type `Object`

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type `Object`

validate (*value*)

validator = <mogwai.properties.validators.RegexValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.String` (**args, **kwargs*)
String/CharField property

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type `basestring | str`

get_default ()

Returns the default value for this graph property if one is available.

Return type `Object | None`

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type `Callable`

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters `value` (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type `Object`

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type `bool`

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if `db_field` is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type `bool`

to_database (*value*)

Converts python value into database value

Return type `Object`

to_python (*value*)

validate (*value*)

validator = `<mogwai.properties.validators.StringValidator object>`

value_manager

alias of `BaseValueManager`

`mogwai.properties.properties.Text`

alias of `String`

class `mogwai.properties.properties.URL` (**args, **kwargs*)

URL Data property type

can_delete

data_type = `'String'`

db_field_name

Returns the name of the mogwai name of this graph property

Return type `basestring | str`

get_default ()

Returns the default value for this graph property if one is available.

Return type `Object | None`

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type `Callable`

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type `Object`

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if db_field is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

Converts python value into database value

Return type Object

to_python (*value*)

Converts data from the database into python values raises a `ValidationError` if the value can't be converted

Return type Object

validate (*value*)

validator = <mogwai.properties.validators.URLValidator object>

value_manager

alias of `BaseValueManager`

class `mogwai.properties.properties.UUID` (*default=<function <lambda>>, **kwargs*)

Universally Unique Identifier (UUID) type - UUID4 by default

can_delete

data_type = 'String'

db_field_name

Returns the name of the mogwai name of this graph property

Return type basestring | str

get_default ()

Returns the default value for this graph property if one is available.

Return type Object | None

get_save_strategy ()

Returns the save strategy attached to this graph property.

Return type Callable

get_value_from_choices (*value, choices*)

Returns the key for the choices tuple of tuples

Note if you are using classes, they must implement the `__in__` and `__eq__` for the logical comparison.

Parameters **value** (*Object*) – The raw value to test if it exists in the valid choices. Could be the key or the value in the dict

Return type Object

has_db_field_prefix

Determines if a field prefix has already been defined.

has_default

Indicates whether or not this graph property has a default value.

Return type bool

instance_counter = 0

set_db_field_prefix (*prefix, override=False*)

Sets the graph property name prefix during document class construction.

set_property_name (*name*)

Sets the graph property name during document class construction.

This value will be ignored if db_field is set in `__init__`

Parameters **name** (*str*) – The name of this graph property

should_save (*first_save=False*)

Indicates whether or not the property should be saved based on it's save strategy.

Return type bool

to_database (*value*)

to_python (*value*)

validate (*value*)

Returns a cleaned and validated value. Raises a `ValidationError` if there's a problem

Return type Object

validator = <mogwai.properties.validators.RegexValidator object>

value_manager

alias of `BaseValueManager`

4.2.6 Relationships

```
class mogwai.relationships.base.Relationship (edge_class, vertex_class, direction='BOTH', strict=True, grem-
lin_path=None, vertex_callback=None,
edge_callback=None, query_callback=None,
create_callback=None)
```

Define incoming and outgoing relationships that exist. Also enforce schema IN, OUT and BOTH directions

Warn if queries return schema violations.

allowed (*edge_type, vertex_type*)

Check whether or not the allowed Edge and Vertex type are compatible with the schema defined

Parameters

- **edge_type** – Edge Class
- **vertex_type** – Vertex Class

Type mogwai.models.Edge

Type mogwai.models.Vertex

Return type bool

create (*args, **kwargs)

Creates a Relationship defined by the schema

Parameters

- **edge_params** (*dict*) – (Optional) Parameters passed to the instantiation method of the Edge
- **vertex_params** (*dict*) – (Optional) Parameters passed to the instantiation method
- **edge_type** (*mogwai.models.Vertex* | *None*) – (Optional) Edge class type, otherwise it defaults to the first Edge type known
- **edge_type** – (Optional) Vertex class type, otherwise it defaults to the first Vertex type known
- **callback** (*method*) – (Optional) Callback function to handle results

Return type tuple(mogwai.models.Edge, mogwai.models.Vertex) | Object

edges (*args, **kwargs)

Query and return all Edges attached to the current Vertex

TODO: fix this, the instance method isn't properly setup :param limit: Limit the number of returned results :type limit: int | long :param offset: Query offset of the number of paginated results :type offset: int | long :param callback: (Optional) Callback function to handle results :type callback: method :rtype: List[mogwai.models.Edge] | Object

query (*args, **kwargs)

Generic Query method for quick access

Parameters

- **edge_types** (*List[mogwai.models.Edge]* | *None*) – List of Edge classes to query against
- **callback** (*method*) – (Optional) Callback function to handle results

Return type mogwai.models.query.Query | Object

vertices (*args, **kwargs)

Query and return all Vertices attached to the current Vertex

TODO: fix this, the instance method isn't properly setup :param limit: Limit the number of returned results :type limit: int | long :param offset: Query offset of the number of paginated results :type offset: int | long :param callback: (Optional) Callback function to handle results :type callback: method :rtype: List[mogwai.models.Vertex] | Object

mogwai.relationships.base.**requires_vertex** (*method*)

4.2.7 Save Strategies

class mogwai.properties.strategy.**SaveAlways**

Save this value every time the corresponding model is saved.

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Save this value every time the corresponding model is saved.

Return type bool

class `mogwai.properties.strategy.SaveOnChange`

Only save this value if it has changed.

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Always save this value if it has changed

Return type `bool`

class `mogwai.properties.strategy.SaveOnDecrease`

Save this value only if it is decreasing

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Only save this value if it is decreasing

Return type `bool`

class `mogwai.properties.strategy.SaveOnIncrease`

Save this value only if it is increasing

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Only save this value if it is increasing

Return type `bool`

class `mogwai.properties.strategy.SaveOnce`

Only save this value once. If it changes throw an exception.

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Always save this value if it has changed

Raises `SaveStrategyException`

Return type `bool`

class `mogwai.properties.strategy.Strategy`

Saving strategies for mogwai. These are used to indicate when a property should be saved after the initial vertex/edge creation.

classmethod condition (*previous_value*, *value*, *has_changed=False*, *first_save=False*,
graph_property=None)

Default save strategy condition

Raises `NotImplementedError`

4.2.8 Property Validators

class `mogwai.properties.validators.BaseValidator` (*message=None*, *code=None*)

code = `u'invalid'`

message = `u'Enter a valid value.'`

class `mogwai.properties.validators.BooleanValidator` (*message=None*, *code=None*)

code = `u'invalid'`

message = `u'Enter a valid Boolean.'`

```
class mogwai.properties.validators.DateTimeUTCValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    message = u'Not a valid UTC DateTime: {}'
```

```
class mogwai.properties.validators.DateTimeValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    message = u'Not a valid DateTime: {}'
```

```
class mogwai.properties.validators.DecimalValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    data_types = (<type 'float'>, <class 'decimal.Decimal'>)
```

```
    message = u'Enter a valid number.'
```

```
class mogwai.properties.validators.DictValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    message = u'Enter a valid dict'
```

```
class mogwai.properties.validators.EmailValidator (regex=None, message=None,
                                                    code=None)
```

```
    code = u'invalid'
```

```
    message = u'Enter a valid email address: {}'
```

```
    regex = <_sre.SRE_Pattern object at 0x2bac8f0>
```

```
class mogwai.properties.validators.FloatValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    data_types = (<type 'float'>,)
```

```
    message = u'Enter a valid number.'
```

```
class mogwai.properties.validators.IntegerValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    data_types = (<type 'int'>, <type 'long'>)
```

```
    message = u'Enter a valid number.'
```

```
class mogwai.properties.validators.ListValidator (message=None, code=None)
```

```
    code = u'invalid'
```

```
    data_types = (<type 'tuple'>, <type 'list'>)
```

```
    message = u'Enter a valid list'
```

```
class mogwai.properties.validators.LongValidator (message=None, code=None)
```

```

code = u'invalid'
data_types = (<type 'int'>, <type 'long'>)
message = u'Enter a valid number.'

```

```
class mogwai.properties.validators.NumericValidator (message=None, code=None)
```

```

code = u'invalid'
data_types = (<type 'float'>, <type 'int'>, <type 'long'>, <class 'decimal.Decimal'>)
message = u'Enter a valid number.'

```

```
class mogwai.properties.validators.PositiveIntegerValidator (message=None,
                                                            code=None)
```

```

code = u'invalid'
data_types = (<type 'int'>, <type 'long'>)
message = u'Enter a valid number.'

```

```
class mogwai.properties.validators.RegexValidator (regex=None,          message=None,
                                                  code=None)
```

```

code = u'invalid'
message = u'Enter a valid value.'
regex = u'

```

```
class mogwai.properties.validators.StringValidator (message=None, code=None)
```

```

code = u'invalid'
data_type = (<type 'basestring'>,)
message = u'Enter a valid string: {}'

```

```
class mogwai.properties.validators.URLValidator (regex=None,          message=None,
                                                code=None)
```

```

code = u'invalid'
message = u'Enter a valid URL address: {}'
regex = <_sre.SRE_Pattern object at 0x2bab530>

```

4.2.9 Metrics

```
class mogwai.metrics.manager.MetricManager
    Manages your Metric Agents
```

Properly inject mogwai metrics, while still allowing custom app metrics

```
count_calls (fn)
    Decorator to track the number of times a function is called.
```

Parameters *fn* (*C{func}*) – the function to be decorated

Returns the decorated function

Return type *C{func}*

counters (*key=None*)

Yield Metric Reporter Counters

Parameters **key** (*basestring*) – The metric key to use

hist_calls (*fn*)

Decorator to check the distribution of return values of a function.

Parameters **fn** (*C{func}*) – the function to be decorated

Returns the decorated function

Return type *C{func}*

histograms (*key=None*)

Yield Metric Reporter Histograms

Parameters **key** (*basestring*) – The metric key to use

meter_calls (*fn*)

Decorator to the rate at which a function is called.

Parameters **fn** (*C{func}*) – the function to be decorated

Returns the decorated function

Return type *C{func}*

meters (*key=None*)

Yield Metric Reporter Meters

Parameters **key** (*basestring*) – The metric key to use

setup_reporters (*metric_reporters*)

Setup the Metric Reporter(s) for the MetricManager

start ()

Start the Metric Reporter

stop ()

Stop the Metric Reporter

time_calls (*fn*)

Decorator to time the execution of the function.

Parameters **fn** (*C{func}*) – the function to be decorated

Returns the decorated function

Return type *C{func}*

timers (*key=None*)

Yield Metric Reporter Timers

Parameters **key** (*basestring*) – The metric key to use

exception `mogwai.metrics.manager.MogwaiMetricsException`

Exception thrown when a metric system error occurs

args

message

class `mogwai.metrics.base.MetricsRegistry` (*clock=<built-in function time>*)

A single interface used to gather metrics on a service. It keeps track of all the relevant Counters, Meters, Histograms, and Timers. It does not have a reference back to its service. The service would create a `L{MetricsRegistry}` to manage all of its metrics tools.

clear ()

counter (*key*)

Gets a counter based on a key, creates a new one if it does not exist.

@param key: name of the metric @type key: C{str}

@return: L{Counter}

dump_metrics ()

Formats all of the metrics and returns them as a dict.

@return: C{list} of C{dict} of metrics

gauge (*key*, *gauge=None*)

get_metrics (*key*)

Gets all the metrics for a specified key.

@param key: name of the metric @type key: C{str}

@return: C{dict}

histogram (*key*)

Gets a histogram based on a key, creates a new one if it does not exist.

@param key: name of the metric @type key: C{str}

@return: L{Histogram}

meter (*key*)

Gets a meter based on a key, creates a new one if it does not exist.

@param key: name of the metric @type key: C{str}

@return: L{Meter}

timer (*key*)

Gets a timer based on a key, creates a new one if it does not exist.

@param key: name of the metric @type key: C{str}

@return: L{Timer}

class `mogwai.metrics.base.RegexRegistry` (*pattern=None*, *clock=<built-in function time>*)

A single interface used to gather metrics on a service. This class uses a regex to combine measures that match a pattern. For example, if you have a REST API, instead of defining a timer for each method, you can use a regex to capture all API calls and group them. A pattern like `‘^/api/(?P<model>)/d+/(?P<verb>)?$’` will group and measure the following:

```
/api/users/1 -> users /api/users/1/edit -> users/edit /api/users/2/edit -> users/edit
```

clear ()

counter (*key*)

dump_metrics ()

Formats all of the metrics and returns them as a dict.

@return: C{list} of C{dict} of metrics

gauge (*key*, *gauge=None*)

get_metrics (*key*)

Gets all the metrics for a specified key.

@param key: name of the metric @type key: C{str}

@return: C{dict}

histogram (*key*)

meter (*key*)

timer (*key*)

exception `mogwai.metrics.base.MogwaiMetricsException`

Exception thrown when a metric system error occurs

args

message

class `mogwai.metrics.base.BaseMetricsReporter` (*registry=None, reportingInterval=300, metric_prefix=None*)

Base Metrics Reporter class

Customize this class for the specific metrics service, ie. Graphite, NewRelic, etc.

get_metrics (*timestamp=None, *args, **kwargs*)

Default pyformance implementation to collect all the metrics from the registries.

Change this if you want to customize for a particular metric collection system, ie. graphite, newrelic, etc.

Parameters **timestamp** (*long | int*) – use this timestamp instead of the generated timestamp when the method is run

Returns Timestamp and aggregated metrics from all registries.

Return type tuple(long | int, dict)

send_metrics (**args, **kwargs*)

Default pyformance implementation is to dump all metrics to STDOUT

Change this if you want to customize for a particular metric collection system, ie. graphite, newrelic, etc.

setup_registry (*registry=None*)

Setup the Metric Reporter with the given registry(s)

start ()

Start the Metric Reporter

start_reporter (*reportingInterval*)

Start the Metric Reporter Agent

Kicks off a loop that every reportingInterval runs the *send_metrics* method.

Parameters **reportingInterval** (*float | long | int*) – The interval (number of seconds) on which to report the collected metrics.

stop ()

Stop the Metric Reporter

`mogwai.metrics.base.ConsoleMetricReporter`

alias of *BaseMetricsReporter*

4.2.10 Compatibility

Utilizing the `six` library for Python 2 and 3 compatibility. More work and testing needs to be done to confirm Python 3 compatibility

4.2.11 Tools

class `mogwai.tools.BlueprintsWrapper` (*class_name=None, setup=None, *args, **kwargs*)
 Abstract implementation for using a Blueprints graph wrapper within a persisted transaction. Within this transaction *g* refers to the wrapped graph.

Parameters

- **class_name** (*str*) – the Blueprints Implementation class name
- **setup** (*str*) – an iterable with additional groovy statements that are executed upon initialization. (optional, default: [])

Returns the used RexPro connection pool (with default session)

Raises `MogwaiBlueprintsWrapperException` if no class name is provided

class `mogwai.tools.Factory`
 Factory for Mogwai models.

ABSTRACT_FACTORY = True

exception `mogwai.tools.ImportStringError` (*import_name, exception*)
 Provides information about a failed `import_string()` attempt.

exception = None

import_name = None

class `mogwai.tools.LazyImportClass` (*import_name*)
 Lazily load and return a class

class

Imports the class and caches

class `mogwai.tools.PartitionGraph` (*write=None, read=None, *args, **kwargs*)
 Wrapper for the Blueprints PartitionGraph, which is documented by <https://github.com/tinkerpop/blueprints/wiki/Partition-Implementation>

Parameters

- **write** (*str*) – the default read+write partition.
- **read** (*iterable*) – optional read partitions.

Returns the used RexPro connection pool (with default session)

Raises `MogwaiBlueprintsWrapperException` if no write partition is provided

class `mogwai.tools.SessionPoolManager` (*bindings=None, pool_size=10*)
 A context manager that exposes a pool whose connections share the same session. If RexPro is used without concurrency, it is not necessary to use connections from the pool explicitly. `connection.execute_query` is patched in this case to use connections from the pool by default. In concurrent mode the pool always needs to be used explicitly, because we cannot rely on global context for patching.

Parameters

- **bindings** (*dict*) – variables that are available throughout the session (optional)
- **pool_size** (*int*) – the maximum number of simultaneous connections for this pool

Returns the used RexPro connection pool (with default session)

class `mogwai.tools.cached_property` (*func, name=None, doc=None*)

A decorator that converts a function into a lazy property. The function wrapped is called the first time to retrieve the result and then that calculated result is used the next time you access the value:

```
class Foo(object):  
  
    @cached_property  
    def foo(self):  
        # calculate something important here  
        return 42
```

The class has to have a `__dict__` in order for this property to work.

`mogwai.tools.import_string(import_name, silent=False)`

Imports an object based on a string. This is useful if you want to use import paths as endpoints or something similar. An import path can be specified either in dotted notation (`xml.sax.saxutils.escape`) or with a colon as object delimiter (`xml.sax.saxutils:escape`).

If `silent` is `True` the return value will be `None` if the import fails.

Parameters

- **import_name** – the dotted name for the object to import.
- **silent** – if set to `True` import errors are ignored and `None` is returned instead.

Returns imported object

`mogwai.spec.get_existing_indices()`

Find all Vertex and Edge types available in the database

`mogwai.spec.write_compiled_indices_to_file(filename, spec=None)`

Write the compile index specification to file

Parameters `filename` (*basestring*) – The file to write to

`mogwai.spec.write_diff_indices_to_file(filename, spec=None)`

Preview of index diff specification to write to file

Parameters `filename` (*basestring*) – The file to write to

`mogwai.spec.write_specs_to_file(filename)`

Generate and write a specification to file

Parameters `filename` (*basestring*) – The file to write to

4.2.12 Exceptions

`mogwai.connection.execute_query(*args, **kwargs)`

Execute a raw Gremlin query with the given parameters passed in.

Parameters

- **query** (*str*) – The Gremlin query to be executed
- **params** (*dict*) – Parameters to the Gremlin query
- **connection** (*RexPro(Sync|Gevent|Eventlet)Connection or None*) – The RexPro connection to execute the query with
- **context** – String context data to include with the query for stats logging

Return type `dict`

`mogwai.connection.generate_spec()`

Generates a titan index and type specification document based on loaded Vertex and Edge models

`mogwai.connection.get_response` (*query, params, isolate, transaction, connection, connection_pool*)

`mogwai.connection.pop_execute_query_kwargs` (*keyword_arguments*)
pop the optional execute query arguments from arbitrary kwargs; return non-None query kwargs in a dict

`mogwai.connection.setup` (*host, graph_name=u'graph', graph_obj_name=u'g', username=u'', password=u'', metric_reporters=None, pool_size=10, concurrency=u'sync'*)
Sets up the connection, and instantiates the models

`mogwai.connection.sync_spec` (*filename, host, graph_name=u'graph', graph_obj_name=u'g', username=u'', password=u'', dry_run=False*)
Sync the given spec file to mogwai.

Parameters

- **filename** (*str*) – The filename of the spec file
- **host** (*str*) – The host the be synced
- **graph_name** (*str*) – The name of the graph to be synced
- **dry_run** (*boolean*) – Only prints generated Gremlin if True

Returns None

4.3 Examples

Common examples for using Mogwai

4.3.1 Quick Start

Usage

Note: This section provides a quick summary of mogwai features. A more detailed listing is available in the full documentation.

Setup Connection

You'll need to setup the connection to the graph database. Mogwai handles connection pooling for you, but you should handle load balancing using dedicated equipment.

```
from mogwai.connection import setup

# Without Authentication
setup('localhost')
# With authentication
#setup('localhost', username='rexster', password='rexster')
# With gevent support
#setup('localhost', concurrency='gevent') # default is Standard Synchronous Python Sockets
# With eventlet support
#setup('localhost', concurrency='eventlet') # default is Standard Synchronous Python Sockets
```

Defining Models

Models declare a Vertex or Edge model and attributes it should possibly contain.

```
from mogwai.models import Vertex, Edge
from mogwai.properties import String, Integer

_val = 0

def counter():
    global _val
    _val += 1
    return _val

class TestVertexModel(Vertex):
    element_type = 'test_vertex_model'

    name = String(default='test_vertex')
    test_val = Integer(default=counter)

class TestEdgeModel(Edge):
    label = 'test_edge_model'

    name = String(default='test_edge')
    test_val = Integer(default=counter)
```

Using Models

mogwai models can be utilized in a similar fashion to the way the Django ORM was constructed, but tailored for Graph Databases.

Creating Vertex or Edge

```
my_vertex_1 = TestVertexModel.create(name='my vertex')
my_vertex_2 = TestVertexModel.create(name='my other vertex')
my_edge = TestEdgeModel.create(outV=my_vertex_1, inV=my_vertex_2, name='my edge')
```

Retrieving a Vertex or Edge

```
# Get all the TestVertexModel Vertices
vertices = TestVertexModel.all()
# Get a subset of vertices by titan ID
vertices = TestVertexModel.all([1234, 5678, 9012])
# Get a vertex by titan ID
vertex = TestVertexModel.get(1234)

# Getting all Edges isn't currently supported
# Get a subset of edges by titan IDs
edges = TestEdgeModel.all(['123-UX4', '215-PX3', '95U-32Z'])
# get a single edge by titan ID
edge = TestEdgeModel.get('123-UX4')
```

```
# Get edge between two vertices
edge = TestEdgeModel.get_between(outV=my_vertex_1, inV=my_vertex_2)
```

Simple Traversals

Vertex Traversals

```
# Get All Edges from the vertex
edges = my_vertex_1.bothE()
# Get outgoing edges from the vertex
edges = my_vertex_1.outE()
# Get incoming edges to the vertex
edges = my_Vertex_1.inE()
# Specify an edge type for any edge traversal operation (works for outE, inE, bothE)
## By using models
test_model_edges = my_vertex_1.outE(TestEdgeModel)
## or by using manual labels
test_model_edges = my_vertex_1.outE('test_edge_model')

# Get all Vertices connected to the vertex
vertices = my_vertex_1.bothV()
# Get all vertices who are connected by edge coming into the current vertex (note uni-directed edges)
vertices = my_vertex_1.outV()
# Get all vertices who are connected by edge coming from the current vertex (note uni-directed edges)
vertices = my_vertex_1.inV()
# Specify an edge type for any edge traversal operation (works for outV, inV, bothV)
## By using models
test_model_vertices = my_vertex_1.outV(TestEdgeModel)
## or by using manual element types
test_model_vertices = my_vertex_1.outV('test_edge_model')
```

Edge Traversals

```
# Get the vertex which is from the outgoing side of the edge
vertex = my_edge.outV()
# Get the vertex which is from the incoming side of the edge
vertex = my_edge.inV()
```

4.3.2 Belongings Example

Here we illustrate creating models and utilizing the Relationship System in parallel with the Vertex Traversal system

```
1 from mogwai.connection import setup
2 from mogwai.models import Vertex, Edge
3 from mogwai import properties
4 from mogwai import relationships
5 from mogwai._compat import print_
6 import datetime
7 from pytz import utc
8 from functools import partial
9
10
11 setup('127.0.0.1')
12
13
14 class OwnsObject (Edge) :
```

```
15
16     label = 'owns_object' # this is optional, will default to the class name
17
18     since = properties.DateTime(required=True,
19                               default=partial(datetime.datetime.now, tz=utc),
20                               description='Owned object since')
21
22
23 class Trinket(Vertex):
24
25     element_type = 'gadget'
26
27     name = properties.String(required=True, max_length=1024)
28
29
30 class Person(Vertex):
31
32     element_type = 'person' # this is optional, will default to the class name
33
34     name = properties.String(required=True, max_length=512)
35     email = properties.Email(required=True)
36
37     # Define a shortcut relationship method
38     belongings = relationships.Relationship(OwnsObject, Trinket)
39
40
41 ## Creation
42 # Create a trinket
43 trinket = Trinket.create(name='Clock')
44
45 # Create a Person
46 bob = Person.create(name='Bob Smith', email='bob@bob.net')
47
48 # Create the Ownership Relationship
49 relationship = OwnsObject.create(outV=bob, inV=trinket)
50
51
52 ## Traversals
53 # Find out what bob owns
54 bob_owns_relationships = bob.belongings.vertices()
55 bob_owns_vertex_traversal = bob.outV(OwnsObject)
56
57 print_("With Relationships: Bob owns: {}".format(bob_owns_relationships))
58
59 # Another method for the same thing
60 print_("With Vertex Traversal: Bob owns: {}".format(bob_owns_vertex_traversal))
61 assert bob_owns_relationships == bob_owns_vertex_traversal
62
63
64 # Find out who owns the trinket
65 print_("With Vertex Traversal: Trinket is owned by: {}".format(trinket.inV(OwnsObject)))
66
67 # When was the trinket owned?
68 print_("Trinket has been owned since: {}".format(relationship.since))
```


4.3.3 Custom Gremlin Example

Here we illustrate creating models and utilizing the Gremlin system for custom methods

Python

```

1 from mogwai.connection import setup
2 from mogwai.models import Vertex, Edge
3 from mogwai import properties
4 from mogwai import gremlin
5 from mogwai._compat import print_
6 import datetime
7 from pytz import utc
8 from functools import partial
9 import os
10
11 setup('127.0.0.1')
12
13
14 class IsFriendsWith(Edge):
15     label = 'is_friends_with'
16
17     since = properties.DateTime(required=True,
18                               default=partial(datetime.datetime.now, tz=utc),
19                               description='Owned object since')
20
21
22 class Person(Vertex):
23
24     element_type = 'person' # this is optional, will default to the class name
25     gremlin_path = os.path.join(os.getcwd(), 'custom_gremlin.groovy')
26
27     name = properties.String(required=True, max_length=512)
28     email = properties.Email(required=True)
29
30     friends_and_friends_of_friends = gremlin.GremlinMethod(method_name='friends_and_friends_of_friends',
31                                                             property=True,
32                                                             defaults={'friend_edge_label': IsFriendsWith})
33     friends_and_friends_of_friends_table = gremlin.GremlinTable(method_name='friends_and_friends_of_friends',
34                                                                    property=True,
35                                                                    defaults={'friend_edge_label': IsFriendsWith})
36
37     @property
38     def friends(self):
39         return self.bothV(IsFriendsWith)
40
41
42 ## Creation
43 # Create the People
44 bob = Person.create(name='Bob', email='bob@bob.net')
45 alice = Person.create(name='Alice', email='alice@alice.net')
46 john = Person.create(name='John', email='john@john.net')
47 tim = Person.create(name='Tim', email='tim@tim.net')
48 kyle = Person.create(name='Kyle', email='kyle@kyle.net')
49
50 # Create Friendships
51 IsFriendsWith.create(outV=bob, inV=alice)

```

```

52 IsFriendsWith.create(outV=alice, inV=john)
53 IsFriendsWith.create(outV=alice, inV=tim)
54 IsFriendsWith.create(outV=tim, inV=kyle)
55
56 ## Traverse
57 # All known direct friends with Bob
58 bobs_friends = bob.friends
59 print_("Bobs direct friends: {}".format(bobs_friends))
60 # Output:
61 # [ Person(element_type=person, id=8132, values={'name': Alice, 'email': alice@alice.net}) ]
62
63
64 # Friends and Friends of Friends using Custom Gremlin Method
65 bobs_friends_and_friends_of_friends = bob.friends_and_friends_of_friends
66 print_("Bobs Friends and Friends of Friends: {}".format(bobs_friends_and_friends_of_friends))
67
68 # Output:
69 # [
70 #   [ Person(element_type=person, id=8128, values={'name': Bob, 'email': bob@bob.net}),
71 #     Person(element_type=person, id=8132, values={'name': Alice, 'email': alice@alice.net}),
72 #     Person(element_type=person, id=8136, values={'name': John, 'email': john@john.net})
73 #   ], [
74 #     Person(element_type=person, id=8128, values={'name': Bob, 'email': bob@bob.net}),
75 #     Person(element_type=person, id=8132, values={'name': Alice, 'email': alice@alice.net}),
76 #     Person(element_type=person, id=8140, values={'name': Tim, 'email': tim@tim.net})
77 #   ]
78 # ]
79
80 for friend_set in bobs_friends_and_friends_of_friends:
81     assert len(friend_set) <= 3
82     assert friend_set[0] == bob
83     assert friend_set[1] == alice
84     assert kyle not in friend_set
85
86
87 # GremlinTable
88 print_("Using GremlinTable:")
89 friend_table = bob.friends_and_friends_of_friends_table
90 for row in friend_table:
91     print_("{} is friends with {}, who is also friends with {}".format(row.person.name, row.friend.na
92
93 # Output:
94 # Using GremlinTable:
95 # Bob is friends with Alice, who is also friends with John
96 # Bob is friends with Alice, who is also friends with Tim

```

Gremlin

custom_gremlin.groovy

```

1 def friends_and_friends_of_friends(id, friend_edge_label) {
2     /**
3     * Traverse a Person and find any Friends of Friends
4     *
5     * :param id: Vertex id of the Person, if null abort
6     * :param friend_edge_label: Edge label of the IsFriendsWith class
7     * :returns: list[Person]

```

```

8     */
9
10    return g.v(id).out(friend_edge_label).loop(1){it.loops < 3}.path
11
12 }
13
14 def friends_and_friends_of_friends_table(id, friend_edge_label) {
15     /**
16      * Traverse a Person and find any Friends of Friends
17      *
18      * :param id: Vertex id of the Person, if null abort
19      * :param friend_edge_label: Edge label of the IsFriendsWith class
20      * :returns: list[Person]
21      */
22     return g.v(id).as('person').out(friend_edge_label).as('friend').out(friend_edge_label).simplePath
23 }

```

4.3.4 Serialization Example

Here we illustrate the serialization of the Vertex and Edge models (Pickle)

```

1  from mogwai.connection import setup
2  from mogwai.models import Vertex, Edge
3  from mogwai import properties
4  from mogwai import relationships
5  from mogwai.compat import print_
6  import datetime
7  from pytz import utc
8  from functools import partial
9  import pickle
10
11
12  setup('127.0.0.1')
13
14
15  class OwnsObject(Edge):
16
17      label = 'owns_object' # this is optional, will default to the class name
18
19      since = properties.DateTime(required=True,
20                                default=partial(datetime.datetime.now, tz=utc),
21                                description='Owned object since')
22
23
24  class Trinket(Vertex):
25
26      element_type = 'gadget'
27
28      name = properties.String(required=True, max_length=1024)
29
30
31  class Person(Vertex):
32
33      element_type = 'person' # this is optional, will default to the class name
34
35      name = properties.String(required=True, max_length=512)
36      email = properties.Email(required=True)

```

```

37
38     # Define a shortcut relationship method
39     belongings = relationships.Relationship(OwnsObject, Trinket)
40
41
42     ## Creation
43     # Create a trinket
44     trinket = Trinket.create(name='Clock')
45
46     # Create a Person
47     bob = Person.create(name='Bob Smith', email='bob@bob.net')
48
49     # Create the Ownership Relationship
50     relationship = OwnsObject.create(outV=bob, inV=trinket)
51
52
53     bob_serialized = pickle.dumps(bob)
54     print_("Bob Serialized: {}".format(bob_serialized))
55     deserialized_bob = pickle.loads(bob_serialized)
56     print_("Bob Deserialized: {}".format(deserialized_bob))
57     assert bob == bob_serialized
58
59     relationship_serialized = pickle.dumps(relationship)
60     print_("Relationship Serialized: {}".format(relationship_serialized))
61     deserialized_relationship = pickle.loads(relationship_serialized)
62     print_("Relationship Deserialized: {}".format(deserialized_relationship))
63     assert relationship == relationship_serialized
64
65
66     trinket_serialized = pickle.dumps(trinket)
67     print_("Trinket Serialized: {}".format(trinket_serialized))
68     deserialized_trinket = pickle.loads(trinket_serialized)
69     print_("Trinket Deserialized: {}".format(deserialized_trinket))
70     assert trinket == trinket_serialized

```

4.3.5 Unknown Properties Example

Here we illustrate the use of un-modeled properties on the Vertex models. Note, this also works for Edge models.

```

1  from mogwai.connection import setup
2  from mogwai.models import Vertex
3  from mogwai import properties
4
5  setup('127.0.0.1')
6
7
8  class Trinket(Vertex):
9
10     element_type = 'gadget'
11
12     name = properties.String(required=True, max_length=1024)
13
14
15     ## Creation
16     # Create a trinket
17     # create with un-modeled property
18     trinket = Trinket.create(name='Clock', engraving='Bob Smith')

```

```

19 assert trinket['engraving'] == 'Bob Smith'
20
21 # get with un-modeled property
22 result = Trinket.get(trinket.id)
23 assert result['engraving'] == 'Bob Smith'
24
25 # change property from trinket (and persist to database)
26 trinket['engraving'] = 'Alice Smith'
27 assert trinket['engraving'] == 'Alice Smith'
28 trinket.save()
29 assert trinket['engraving'] == 'Alice Smith'
30
31 # delete property from trinke (and persist to database)
32 del trinket['engraving']
33 try:
34     result = trinket['engraving']
35     raise Exception("Property shouldn't exist")
36 except AttributeError:
37     pass
38 trinket.save()
39 try:
40     result = trinket['engraving']
41     raise Exception("Property shouldn't exist")
42 except AttributeError:
43     pass
44
45 # iterate through keys
46 keys = [key for key in trinket]
47 keys = [key for key in trinket.keys()]
48 values = [value for value in trinket.values()]
49 items = [item for item in trinket.items()]

```

4.3.6 Groovy Imports Example

Here we illustrate the use of importing in the groovy file, which applies these imports to all the function definitions.

```

1 import com.thinkaurelius.titan.core.util.*;
2
3 def test_method(a1, a2) {
4     return a1, a2
5 }
6
7 def test_second_method(a1, a2) {
8     return (a1..a2)
9 }

```

Here we illustrate the use of additional imports directly on specific GremlinMethod or GremlinValue:

```

1 from mogwai.connection import setup
2 from mogwai.models import Vertex
3 from mogwai import properties
4 from mogwai.gremlin import GremlinMethod
5
6 setup('127.0.0.1')
7
8
9 class Trinket(Vertex):

```

```

10
11     element_type = 'gadget'
12
13     name = properties.String(required=True, max_length=1024)
14
15     test_method = GremlinMethod(path='example_groovy_imports.groovy', method_name='test_method',
16                                 imports=['com.thinkaurelius.titan.core.util.*'], classmethod=True)
17
18     # Call the test method:
19     result1, result2 = Trinket.test_method(1, 2)
20     assert result1 == 1
21     assert result2 == 2

```

4.3.7 Session Management Example

Session Wrapping

The default connection pool that is used to execute queries uses a new session for every request. This fully isolates queries from each other, which is a good idea in most cases. There are, however, also scenarios in which multiple queries need to share some kind of state. This can be done with a special kind of connection pool, which by default spawns connections that use the same session.

```

1  from mogwai import connection, properties
2  from mogwai.models import Vertex, Edge
3  from mogwai.tools import SessionPoolManager, BlueprintsWrapper, PartitionGraph
4  from mogwai.exceptions import MogwaiGremlinException
5
6  connection.setup('localhost')
7
8  ##
9  # Persist a session with SessionPoolManager
10 ##
11
12 k = 10
13
14 with SessionPoolManager(bindings={'k': k}):
15     gsk = connection.execute_query("powers of ${k}")
16     pysk = "powers of {}".format(k)
17     assert gsk == pysk
18
19     kk = connection.execute_query("k * k")
20     assert kk == k * k
21
22
23 ##
24 # Wrap the graph with a Blueprints Implementation
25 ##
26
27 class BlueprintsWrapperVertex(Vertex):
28     element_type = 'blueprints_wrapper_vertex'
29     name = properties.String(required=True, max_length=128)
30
31
32 class BlueprintsWrapperEdge(Edge):
33     element_type = 'blueprints_wrapper_edge'
34     name = properties.String(required=True, max_length=128)

```

```

35
36 v0 = BlueprintsWrapperVertex.create(name="valid")
37
38 with BlueprintsWrapper(class_name='ReadOnlyGraph'):
39     v0 = BlueprintsWrapperVertex.get(v0._id)
40     try:
41         BlueprintsWrapperVertex.create(name="illegal")
42     except MogwaiGremlinException:
43         print "java.lang.UnsupportedOperationException: \
44             It is not possible to mutate a ReadOnlyGraph"
45
46
47 ##
48 # Treat the graph as a Partition Graph
49 ##
50
51 with PartitionGraph(write='a'):
52     v1 = BlueprintsWrapperVertex.create(name="only in a")
53     v3 = BlueprintsWrapperVertex.create(name="started in a")
54
55 with PartitionGraph(write='b', read=['a']):
56     v2 = BlueprintsWrapperVertex.create(name="only in b")
57     e1 = BlueprintsWrapperEdge.create(outV=v2, inV=v1, name="only in b")
58     v3.name = "still in a"
59     v3.save()
60
61 with PartitionGraph(write='a'):
62     v1 = BlueprintsWrapperVertex.get(v1._id)
63     assert len(v1.bothE()) == 0
64     try:
65         BlueprintsWrapperVertex.get(v2._id)
66     except BlueprintsWrapperVertex.DoesNotExist:
67         print "vertex is located in partition B"
68
69 # outside of the session all the elements are accessible
70 print v1.bothE()[0]
71 # BlueprintsWrapperEdge(label=blueprints_wra..., values={'name': 'only in b'})
72
73 print dict(BlueprintsWrapperVertex.get(v3._id))
74 # {'_partition': 'a', 'name': 'still in a'}

```

Concurrent Session Wrapping

Note: The pool needs to be referenced explicitly by querying methods in the wrapper when the connection to Rexster is set up with e.g. *eventlet* or *gevent*.

The type of concurrency that is available in mogwai and rexpro is based on coroutines, which share the same global context. It would thus be unsafe to keep the session or its associated pool in a global variable. This would cause concurrently executed session wrappers to change the session key for all ongoing requests, also for those that should be executed within a different session.

All graph element methods that perform operations on the graph accept the optional keyword arguments *transaction*, *isolate*, and *pool*, which can override the defaults set by *execute_query*.

execute_query(query, params={}, transaction=True, isolate=True, pool=None

The following examples use *eventlet* for concurrent querying:

```

1 from mogwai import connection, properties
2 from mogwai.models import Vertex, Edge
3 from mogwai.tools import SessionPoolManager, PartitionGraph
4 import eventlet
5
6 connection.setup('localhost', concurrency='eventlet')
7
8 ##
9 # Persist a session with SessionPoolManager
10 ##
11
12 def isolation_test(scope):
13     wrapper_config = {
14         'bindings': {'scope': scope},
15         'pool_size': 5
16     }
17     scope_values = []
18
19     with SessionPoolManager(**wrapper_config) as pool:
20         for i in range(7):
21             scope_val = connection.execute_query(
22                 "scope *= 2",
23                 isolate=False,
24                 pool=pool
25             )
26             scope_values.append(scope_val)
27
28     return scope, scope_values
29
30 pile = eventlet.GreenPile()
31 [pile.spawn(isolation_test, i) for i in range(10)]
32
33 for scope, scope_values in pile:
34     assert scope_values == [scope * 2**i for i in range(1, 8)]
35
36
37 ##
38 # Treat the graph as a Partition Graph
39 ##
40
41 class BlueprintsWrapperVertex(Vertex):
42     element_type = 'blueprints_wrapper_vertex'
43     name = properties.String(required=True, max_length=128)
44
45
46 class BlueprintsWrapperEdge(Edge):
47     element_type = 'blueprints_wrapper_edge'
48     name = properties.String(required=True, max_length=128)
49
50 n = 5
51
52 with PartitionGraph(write='a') as pool:
53     av_pile = eventlet.GreenPile()
54     for i in range(n):
55         av_pile.spawn(BlueprintsWrapperVertex.create, name="only in a", pool=pool)
56     vertices_in_a = list(av_pile)
57
58 with PartitionGraph(write='b', read=['a']) as pool:

```



```

59  bv_pile = eventlet.GreenPile()
60  for i in range(n):
61      bv_pile.spawn(BlueprintsWrapperVertex.create, name="only in b", pool=pool)
62
63  [pile.spawn(isolation_test, i) for i in range(n)]
64
65  be_pile = eventlet.GreenPile()
66  for vb in bv_pile:
67      va = vertices_in_a.pop()
68      scope, scope_values = pile.next()
69      va[str(scope)] = scope_values
70      av_pile.spawn(va.save, pool=pool)
71
72      be_pile.spawn(BlueprintsWrapperEdge.create, outV=vb, inV=va,
73                  name="only in b", pool=pool)
74
75  vertices_in_a = [av for av in av_pile]
76  edges_in_b = [be for be in be_pile]
77
78  for v in BlueprintsWrapperVertex.all():
79      print v._id, dict(v), v.outE()
80
81  [pile.spawn(v.delete) for v in BlueprintsWrapperVertex.all()]
82  list(pile)

```

4.4 ChangeLog

Changes to the library are recorded here.

4.4.1 v0.7.5

- Loosen six version requirement, reflected updated dependency for loosened rexpri version

4.4.2 v0.7.4

- Fix minor bug with Edge.outV not returning Vertex - thanks to Leif Asgeirsson

4.4.3 v0.7.3

- Remove trace messages and clean-up developer logs

4.4.4 v0.7.1

- Fixed Rexpri Connection Pooling connection soft-closing on exception
- Fixed Relationship isolation bug (issue #10)

4.4.5 v0.7.0

- BluePrints PartitionGraph support - thanks to Alex Olieman (aolieman)

4.4.6 v0.6.8

- critical Relationships bugfix

4.4.7 v0.6.7

- fixed GremlinTable method to return a new Table object that composes Row(s)

4.4.8 v0.6.6

- find_by_value bugfix - thanks to Justin Hohner for identifying this bug.

4.4.9 v0.6.5

- Proper connection closing bugfix - thanks to Elizabeth Ramirez

4.4.10 v0.6.4

- Use blueprints element.setProperty method instead of deprecated addProperty - thanks to Kaisen Lin

4.4.11 v0.6.3

- as_dict() method now includes element id

4.4.12 v0.6.2

- Sane default __repr__ for Edges, feature parity to Vertex

4.4.13 v0.6.1

- Save Strategy bugfix - thanks to Nick Vollmar

4.4.14 v0.6.0

- Titan 0.5.0 support confirmed
- Tests updated to reflect changes, do note running tests against this requires titan 0.5.0

4.4.15 v0.5.0

- Python 3.3 and 3.4 compatibility - requires rexpro 0.3.0

4.4.16 v0.4.3

- Added support for groovy imports on file level, as well as through GremlinMethod/GremlinValue (*See Example*)

4.4.17 v0.4.2

- Added support for unknown properties and dictionary-like access (*See Example*)

4.4.18 v0.4.1

- Missing import fix
- setup.py typo fix

4.4.19 v0.4.0

- Support concurrent connections via gevent and eventlet through rexprou support in the 0.2.0 release
- All Properties support None when not required. Be aware when developing, especially around the Boolean property, since, when not required, it can actually be in 3 states (True, False and None) or (true, false and null in groovy)

4.4.20 v0.3.3

- Fixed and improved find_by_value for both Edge and Vertex

4.4.21 v0.3.2

- Fixed Circular Import
- Fixed DateTime UTC bug
- Wercker Integration
- Documentation updates

4.4.22 v0.3.1

- Bug Fixes
- Documentation updates

4.4.23 v0.3.0

- Utilize new rexprou.RexProConnectionPool
- Includes new rexprou green-thread friendly gevent.socket RexProSockets

4.4.24 v0.2.13

- setup.py install_requires hot fix

4.4.25 v0.2.12

- Public CI preview

4.4.26 v0.2.11

- Documentation Updates

4.4.27 v0.2.10

- Minor bug fixes

4.4.28 v0.2.9

Serializable models via pickle.

```
import pickle

vertex = MyTestVertex.create(name='test')
serialized_vertex = pickle.dumps(vertex)
deserialized_vertex = pickle.loads(serialized_vertex)
assert vertex == deserialized_vertex
```

4.4.29 v0.2.8

Re-Release of Mogwai to the public. Name change to Mogwai, which loosely means “gremlin”. This is a major refactor of the original *thunderdome* library by Blake.

- Using RexPro, updated library to utilize RexPro and compatible with Titan 0.4.2
- Refactored library, changed the way properties are handled, validated and their associated save strategies.
- Removed vid and eid as primary keys, titan generates unique primary keys that we can utilize. Now accessible via Element._id or Element.id (the latter is a property shortcut to Element._id)
- Added groovy tests, updated gremlin base method for new _type_name
- Added interactive shell with some slight magic:

```
Usage:
  python -m mogwai.shell --host localhost
For more help see:
  python -m mogwai.shell --help
Also HELP is available in the shell
```

- Preview of index specification system, initial commit
- Relationship system, includes generic query method, create relationship method and strict relationship checker
- Fixed groovy files to only use local variables in core structure, will prevent Concurrent Global variable scope locks
- Special Enum Vertex metaclass now available. ie. *MyVertex.MY_ENUM* will be able to resolve to an actual database entry
- Performance monitoring tools now available - Customizable for different metric reporting mechanisms, ie, console, logs, graphite, newrelic.
- Apache 2.0 License

4.5 Contributing

4.5.1 License

mogwai is distributed under the [Apache 2.0 License](#).

4.5.2 Issues

Issues should be opened through [Bitbucket Issues](#); whenever possible, a pull request should be included.

4.5.3 Pull Requests

General Rules:

- All Tests must pass
- Coverage shouldn't decrease
- All Pull Requests should be rebased against master **before** submitting the PR.

All pull requests should pass the test suite, which can be launched simply with:

```
$ make test
```

Note: Running test requires *nosetests*, *coverage*, *six*, *pyformance*, *rexpro* and *factory_boy*. As well an available titan server.

In order to test coverage, please use:

```
$ pip install coverage
$ coverage erase; make coverage
```

4.5.4 Test Server

mogwai was designed for [Titan](#), other graph databases that utilize [Blueprints](#) may be compatible, but further testing would be needed.

Currently Titan 0.4.4 is known to work and can be downloaded: [Titan](#).

4.6 Ideas

We'll keep a running list of ideas for future enhancements. Please note this is likely incomplete and not always up-to-date with the issue tracker. Also these are in no-particular order.

- **Titan 0.5.x + support → TinkerPop3 and Gremlin3 Support**
 - Websocket support for streaming results (generators)
 - Build in pluggable gevent/eventlet support for streaming results – mogwai operations should be asynchronous
- **Improved Index/Constraint Specification System**
 - Create Faunus or runnable job against the database to enforce a specification

- **Support Migrations - The 'South' project lends inspiration here.**
- **Add Relationship attribute magic**
 - Schema enforcement for Faunus
 - Creation/Traversal enforcement
- **Integrate additional common metric collectors**
- **Filter lazy-evaluated gremlin query.**
 - Perhaps: `MyVertexModel.query().filter(someval__lte=2)`
- **Database Export/Import functionality**
- **Enum MetaClass optional memcache/etc support.**
- **Gremlin Query builder** * Optional groovy script generator based on query.
- **Groovy script inspections tool for optimization hints, common algorithms, possible syntax errors.**

Indices and tables

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

m

`mogwai.connection`, 56
`mogwai.gremlin.base`, 18
`mogwai.gremlin.groovy`, 20
`mogwai.gremlin.table`, 21
`mogwai.metrics.base`, 52
`mogwai.metrics.manager`, 51
`mogwai.models.edge`, 16
`mogwai.models.vertex`, 12
`mogwai.properties.base`, 22
`mogwai.properties.properties`, 24
`mogwai.properties.strategy`, 48
`mogwai.properties.validators`, 49
`mogwai.relationships.base`, 47
`mogwai.spec`, 56
`mogwai.tools`, 55

A

ABSTRACT_FACTORY (mogwai.tools.Factory attribute), 55

all() (mogwai.models.edge.Edge class method), 16

all() (mogwai.models.vertex.Vertex class method), 13

allowed() (mogwai.relationships.base.Relationship method), 47

args (mogwai.gremlin.groovy.GroovyFunction attribute), 20

args (mogwai.metrics.base.MogwaiMetricsException attribute), 54

args (mogwai.metrics.manager.MogwaiMetricsException attribute), 52

args (mogwai.models.edge.Edge.DoesNotExist attribute), 16

args (mogwai.models.edge.Edge.MultipleObjectsReturned attribute), 16

args (mogwai.models.edge.Edge.WrongElementType attribute), 16

args (mogwai.models.vertex.Vertex.DoesNotExist attribute), 12

args (mogwai.models.vertex.Vertex.MultipleObjectsReturned attribute), 12

args (mogwai.models.vertex.Vertex.WrongElementType attribute), 12

as_dict() (mogwai.models.edge.Edge method), 16

as_dict() (mogwai.models.vertex.Vertex method), 13

as_save_params() (mogwai.models.edge.Edge method), 16

as_save_params() (mogwai.models.vertex.Vertex method), 13

B

BaseGremlinMethod (class in mogwai.gremlin.base), 18

BaseMetricsReporter (class in mogwai.metrics.base), 54

BaseValidator (class in mogwai.properties.validators), 49

BaseValueManager (class in mogwai.properties.base), 22

BlueprintsWrapper (class in mogwai.tools), 55

body (mogwai.gremlin.groovy.GroovyFunction attribute), 20

Boolean (class in mogwai.properties.properties), 24

BooleanValidator (class in mogwai.properties.validators), 49

bothE() (mogwai.models.vertex.Vertex method), 13

bothV() (mogwai.models.vertex.Vertex method), 13

C

cached_property (class in mogwai.tools), 55

can_delete (mogwai.properties.base.GraphProperty attribute), 22

can_delete (mogwai.properties.properties.Boolean attribute), 24

can_delete (mogwai.properties.properties.DateTime attribute), 25

can_delete (mogwai.properties.properties.DateTimeNaive attribute), 26

can_delete (mogwai.properties.properties.Decimal attribute), 27

can_delete (mogwai.properties.properties.Dictionary attribute), 28

can_delete (mogwai.properties.properties.Double attribute), 29

can_delete (mogwai.properties.properties.Email attribute), 30

can_delete (mogwai.properties.properties.Float attribute), 31

can_delete (mogwai.properties.properties.Integer attribute), 36

can_delete (mogwai.properties.properties.IPV4 attribute), 32

can_delete (mogwai.properties.properties.IPV6 attribute), 34

can_delete (mogwai.properties.properties.IPV6WithV4 attribute), 35

can_delete (mogwai.properties.properties.List attribute), 37

can_delete (mogwai.properties.properties.Long attribute), 38

can_delete (mogwai.properties.properties.PositiveInteger attribute), 39

- can_delete (mogwai.properties.properties.PositiveLong attribute), 41
- can_delete (mogwai.properties.properties.Short attribute), 42
- can_delete (mogwai.properties.properties.Slug attribute), 43
- can_delete (mogwai.properties.properties.String attribute), 44
- can_delete (mogwai.properties.properties.URL attribute), 45
- can_delete (mogwai.properties.properties.UUID attribute), 46
- changed (mogwai.properties.base.BaseValueManager attribute), 22
- clear() (mogwai.metrics.base.MetricsRegistry method), 52
- clear() (mogwai.metrics.base.RegexRegistry method), 53
- code (mogwai.properties.validators.BaseValidator attribute), 49
- code (mogwai.properties.validators.BooleanValidator attribute), 49
- code (mogwai.properties.validators.DateTimeUTCValidator attribute), 50
- code (mogwai.properties.validators.DateTimeValidator attribute), 50
- code (mogwai.properties.validators.DecimalValidator attribute), 50
- code (mogwai.properties.validators.DictValidator attribute), 50
- code (mogwai.properties.validators.EmailValidator attribute), 50
- code (mogwai.properties.validators.FloatValidator attribute), 50
- code (mogwai.properties.validators.IntegerValidator attribute), 50
- code (mogwai.properties.validators.ListValidator attribute), 50
- code (mogwai.properties.validators.LongValidator attribute), 50
- code (mogwai.properties.validators.NumericValidator attribute), 51
- code (mogwai.properties.validators.PositiveIntegerValidator attribute), 51
- code (mogwai.properties.validators.RegexValidator attribute), 51
- code (mogwai.properties.validators.StringValidator attribute), 51
- code (mogwai.properties.validators.URLValidator attribute), 51
- comment_list (mogwai.gremlin.groovy.GroovyImport attribute), 21
- CommentVar (mogwai.gremlin.groovy.GroovyImportParser attribute), 21
- condition() (mogwai.properties.strategy.SaveAlways class method), 48
- condition() (mogwai.properties.strategy.SaveOnce class method), 49
- condition() (mogwai.properties.strategy.SaveOnChange class method), 49
- condition() (mogwai.properties.strategy.SaveOnDecrease class method), 49
- condition() (mogwai.properties.strategy.SaveOnIncrease class method), 49
- condition() (mogwai.properties.strategy.Strategy class method), 49
- configure_method() (mogwai.gremlin.base.BaseGremlinMethod method), 18
- configure_method() (mogwai.gremlin.base.GremlinMethod method), 19
- configure_method() (mogwai.gremlin.base.GremlinTable method), 19
- configure_method() (mogwai.gremlin.base.GremlinValue method), 20
- ConsoleMetricReporter (in module mogwai.metrics.base), 54
- count() (mogwai.gremlin.groovy.GroovyFunction method), 20
- count() (mogwai.gremlin.groovy.GroovyImport method), 21
- count_calls() (mogwai.metrics.manager.MetricManager method), 51
- counter() (mogwai.metrics.base.MetricsRegistry method), 53
- counter() (mogwai.metrics.base.RegexRegistry method), 53
- counters() (mogwai.metrics.manager.MetricManager method), 51
- create() (mogwai.models.edge.Edge class method), 16
- create() (mogwai.models.vertex.Vertex method), 13
- create() (mogwai.relationships.base.Relationship method), 48
- ## D
- data_type (mogwai.properties.base.GraphProperty attribute), 22
- data_type (mogwai.properties.properties.Boolean attribute), 24
- data_type (mogwai.properties.properties.DateTime attribute), 25
- data_type (mogwai.properties.properties.DateTimeNaive attribute), 26
- data_type (mogwai.properties.properties.Decimal attribute), 27
- data_type (mogwai.properties.properties.Dictionary attribute), 28

- data_type (mogwai.properties.properties.Double attribute), 29
- data_type (mogwai.properties.properties.Email attribute), 30
- data_type (mogwai.properties.properties.Float attribute), 31
- data_type (mogwai.properties.properties.Integer attribute), 36
- data_type (mogwai.properties.properties.IPV4 attribute), 32
- data_type (mogwai.properties.properties.IPV6 attribute), 34
- data_type (mogwai.properties.properties.IPV6WithV4 attribute), 35
- data_type (mogwai.properties.properties.List attribute), 37
- data_type (mogwai.properties.properties.Long attribute), 38
- data_type (mogwai.properties.properties.PositiveInteger attribute), 39
- data_type (mogwai.properties.properties.PositiveLong attribute), 41
- data_type (mogwai.properties.properties.Short attribute), 42
- data_type (mogwai.properties.properties.Slug attribute), 43
- data_type (mogwai.properties.properties.String attribute), 44
- data_type (mogwai.properties.properties.URL attribute), 45
- data_type (mogwai.properties.properties.UUID attribute), 46
- data_type (mogwai.properties.validators.StringValidator attribute), 51
- data_types (mogwai.properties.validators.DecimalValidator attribute), 50
- data_types (mogwai.properties.validators.FloatValidator attribute), 50
- data_types (mogwai.properties.validators.IntegerValidator attribute), 50
- data_types (mogwai.properties.validators.ListValidator attribute), 50
- data_types (mogwai.properties.validators.LongValidator attribute), 51
- data_types (mogwai.properties.validators.NumericValidator attribute), 51
- data_types (mogwai.properties.validators.PositiveIntegerValidator attribute), 51
- DateTime (class in mogwai.properties.properties), 25
- DateTimeNaive (class in mogwai.properties.properties), 26
- DateTimeUTCValidator (class in mogwai.properties.validators), 49
- DateTimeValidator (class in mogwai.properties.validators), 50
- db_field_name (mogwai.properties.base.GraphProperty attribute), 22
- db_field_name (mogwai.properties.properties.Boolean attribute), 24
- db_field_name (mogwai.properties.properties.DateTime attribute), 25
- db_field_name (mogwai.properties.properties.DateTimeNaive attribute), 26
- db_field_name (mogwai.properties.properties.Decimal attribute), 27
- db_field_name (mogwai.properties.properties.Dictionary attribute), 28
- db_field_name (mogwai.properties.properties.Double attribute), 29
- db_field_name (mogwai.properties.properties.Email attribute), 30
- db_field_name (mogwai.properties.properties.Float attribute), 31
- db_field_name (mogwai.properties.properties.Integer attribute), 36
- db_field_name (mogwai.properties.properties.IPV4 attribute), 32
- db_field_name (mogwai.properties.properties.IPV6 attribute), 34
- db_field_name (mogwai.properties.properties.IPV6WithV4 attribute), 35
- db_field_name (mogwai.properties.properties.List attribute), 37
- db_field_name (mogwai.properties.properties.Long attribute), 38
- db_field_name (mogwai.properties.properties.PositiveInteger attribute), 39
- db_field_name (mogwai.properties.properties.PositiveLong attribute), 41
- db_field_name (mogwai.properties.properties.Short attribute), 42
- db_field_name (mogwai.properties.properties.Slug attribute), 43
- db_field_name (mogwai.properties.properties.String attribute), 44
- db_field_name (mogwai.properties.properties.URL attribute), 45
- db_field_name (mogwai.properties.properties.UUID attribute), 46
- Decimal (class in mogwai.properties.properties), 27
- DecimalValidator (class in mogwai.properties.validators), 50
- defn (mogwai.gremlin.groovy.GroovyFunction attribute), 20
- delete() (mogwai.models.edge.Edge method), 17
- delete() (mogwai.models.vertex.Vertex method), 13
- delete_inE() (mogwai.models.vertex.Vertex method), 13

delete_inV() (mogwai.models.vertex.Vertex method), 13
 delete_outE() (mogwai.models.vertex.Vertex method), 13
 delete_outV() (mogwai.models.vertex.Vertex method), 13
 deleted (mogwai.properties.base.BaseValueManager attribute), 22
 delval() (mogwai.properties.base.BaseValueManager method), 22
 deserialize() (mogwai.models.edge.Edge method), 17
 deserialize() (mogwai.models.vertex.Vertex method), 14
 Dictionary (class in mogwai.properties.properties), 28
 DictValidator (class in mogwai.properties.validators), 50
 Double (class in mogwai.properties.properties), 29
 dump_metrics() (mogwai.metrics.base.MetricsRegistry method), 53
 dump_metrics() (mogwai.metrics.base.RegexRegistry method), 53

E

Edge (class in mogwai.models.edge), 16
 Edge.DoesNotExist, 16
 Edge.MultipleObjectsReturned, 16
 Edge.WrongElementType, 16
 EdgeMetaClass (class in mogwai.models.edge), 18
 edges() (mogwai.relationships.base.Relationship method), 48
 element_type (mogwai.models.vertex.Vertex attribute), 14
 Email (class in mogwai.properties.properties), 30
 EmailValidator (class in mogwai.properties.validators), 50
 enums (mogwai.models.vertex.EnumVertexBaseMeta attribute), 12
 EnumVertexBaseMeta (class in mogwai.models.vertex), 12
 exception (mogwai.tools.ImportStringError attribute), 55
 execute_query() (in module mogwai.connection), 11, 56

F

Factory (class in mogwai.tools), 55
 FACTORY_CLASS (mogwai.models.edge.Edge attribute), 16
 FACTORY_CLASS (mogwai.models.vertex.Vertex attribute), 12
 find_by_value() (mogwai.models.edge.Edge class method), 17
 find_by_value() (mogwai.models.vertex.Vertex class method), 14
 Float (class in mogwai.properties.properties), 31
 FloatValidator (class in mogwai.properties.validators), 50
 FuncDefn (mogwai.gremlin.groovy.GroovyFunctionParser attribute), 20
 FuncName (mogwai.gremlin.groovy.GroovyFunctionParser attribute), 20

G

gauge() (mogwai.metrics.base.MetricsRegistry method), 53
 gauge() (mogwai.metrics.base.RegexRegistry method), 53
 generate_spec() (in module mogwai.connection), 11, 56
 get() (mogwai.models.edge.Edge class method), 17
 get() (mogwai.models.vertex.Vertex class method), 14
 get_between() (mogwai.models.edge.Edge class method), 17
 get_default() (mogwai.properties.base.GraphProperty method), 23
 get_default() (mogwai.properties.properties.Boolean method), 24
 get_default() (mogwai.properties.properties.DateTime method), 25
 get_default() (mogwai.properties.properties.DateTimeNaive method), 26
 get_default() (mogwai.properties.properties.Decimal method), 27
 get_default() (mogwai.properties.properties.Dictionary method), 28
 get_default() (mogwai.properties.properties.Double method), 29
 get_default() (mogwai.properties.properties.Email method), 30
 get_default() (mogwai.properties.properties.Float method), 32
 get_default() (mogwai.properties.properties.Integer method), 36
 get_default() (mogwai.properties.properties.IPV4 method), 33
 get_default() (mogwai.properties.properties.IPV6 method), 34
 get_default() (mogwai.properties.properties.IPV6WithV4 method), 35
 get_default() (mogwai.properties.properties.List method), 37
 get_default() (mogwai.properties.properties.Long method), 38
 get_default() (mogwai.properties.properties.PositiveInteger method), 40
 get_default() (mogwai.properties.properties.PositiveLong method), 41
 get_default() (mogwai.properties.properties.Short method), 42
 get_default() (mogwai.properties.properties.Slug method), 43
 get_default() (mogwai.properties.properties.String method), 44
 get_default() (mogwai.properties.properties.URL method), 45
 get_default() (mogwai.properties.properties.UUID method), 46

- get_element_type() (mogwai.models.vertex.Vertex class method), 14
- get_existing_indices() (in module mogwai.spec), 56
- get_label() (mogwai.models.edge.Edge class method), 17
- get_metrics() (mogwai.metrics.base.BaseMetricsReporter method), 54
- get_metrics() (mogwai.metrics.base.MetricsRegistry method), 53
- get_metrics() (mogwai.metrics.base.RegexRegistry method), 53
- get_property() (mogwai.properties.base.BaseValueManager method), 22
- get_property_by_name() (mogwai.models.edge.Edge method), 17
- get_property_by_name() (mogwai.models.vertex.Vertex method), 14
- get_response() (in module mogwai.connection), 11, 56
- get_save_strategy() (mogwai.properties.base.GraphProperty method), 23
- get_save_strategy() (mogwai.properties.properties.Boolean method), 24
- get_save_strategy() (mogwai.properties.properties.DateTime method), 25
- get_save_strategy() (mogwai.properties.properties.DateTimeNaive method), 26
- get_save_strategy() (mogwai.properties.properties.Decimal method), 27
- get_save_strategy() (mogwai.properties.properties.Dictionary method), 28
- get_save_strategy() (mogwai.properties.properties.Double method), 29
- get_save_strategy() (mogwai.properties.properties.Email method), 30
- get_save_strategy() (mogwai.properties.properties.Float method), 32
- get_save_strategy() (mogwai.properties.properties.Integer method), 36
- get_save_strategy() (mogwai.properties.properties.IPV4 method), 33
- get_save_strategy() (mogwai.properties.properties.IPV6 method), 34
- get_save_strategy() (mogwai.properties.properties.IPV6WithV4 method), 35
- get_save_strategy() (mogwai.properties.properties.List method), 37
- get_save_strategy() (mogwai.properties.properties.Long method), 38
- get_save_strategy() (mogwai.properties.properties.PositiveInteger method), 40
- get_save_strategy() (mogwai.properties.properties.PositiveLong method), 41
- get_save_strategy() (mogwai.properties.properties.Short method), 42
- get_save_strategy() (mogwai.properties.properties.Slug method), 43
- get_save_strategy() (mogwai.properties.properties.String method), 44
- get_save_strategy() (mogwai.properties.properties.URL method), 45
- get_save_strategy() (mogwai.properties.properties.UUID method), 46
- get_value_from_choices() (mogwai.properties.base.GraphProperty class method), 23
- get_value_from_choices() (mogwai.properties.properties.Boolean method), 24
- get_value_from_choices() (mogwai.properties.properties.DateTime method), 25
- get_value_from_choices() (mogwai.properties.properties.DateTimeNaive method), 26
- get_value_from_choices() (mogwai.properties.properties.Decimal method), 27
- get_value_from_choices() (mogwai.properties.properties.Dictionary method), 28
- get_value_from_choices() (mogwai.properties.properties.Double method), 29
- get_value_from_choices() (mogwai.properties.properties.Email method), 31
- get_value_from_choices() (mogwai.properties.properties.Float method), 32
- get_value_from_choices() (mogwai.properties.properties.Integer method), 36
- get_value_from_choices() (mogwai.properties.properties.IPV4 method), 33
- get_value_from_choices() (mogwai.properties.properties.IPV6 method), 34

- [get_value_from_choices\(\)](#) (mogwai.properties.properties.IPV6WithV4 method), 35
[get_value_from_choices\(\)](#) (mogwai.properties.properties.List method), 37
[get_value_from_choices\(\)](#) (mogwai.properties.properties.Long method), 38
[get_value_from_choices\(\)](#) (mogwai.properties.properties.PositiveInteger method), 40
[get_value_from_choices\(\)](#) (mogwai.properties.properties.PositiveLong method), 41
[get_value_from_choices\(\)](#) (mogwai.properties.properties.Short method), 42
[get_value_from_choices\(\)](#) (mogwai.properties.properties.Slug method), 43
[get_value_from_choices\(\)](#) (mogwai.properties.properties.String method), 44
[get_value_from_choices\(\)](#) (mogwai.properties.properties.URL method), 45
[get_value_from_choices\(\)](#) (mogwai.properties.properties.UUID method), 46
[getval\(\)](#) (mogwai.properties.base.BaseValueManager method), 22
[GraphProperty](#) (class in mogwai.properties.base), 22
[gremlin_path](#) (mogwai.models.edge.Edge attribute), 17
[gremlin_path](#) (mogwai.models.vertex.Vertex attribute), 14
[GremlinMethod](#) (class in mogwai.gremlin.base), 19
[GremlinTable](#) (class in mogwai.gremlin.base), 19
[GremlinValue](#) (class in mogwai.gremlin.base), 19
[GroovyFileDef](#) (in module mogwai.gremlin.groovy), 20
[GroovyFunction](#) (class in mogwai.gremlin.groovy), 20
[GroovyFunctionParser](#) (class in mogwai.gremlin.groovy), 20
[GroovyImport](#) (class in mogwai.gremlin.groovy), 20
[GroovyImportParser](#) (class in mogwai.gremlin.groovy), 21

H

[has_db_field_prefix](#) (mogwai.properties.base.GraphProperty attribute), 23
[has_db_field_prefix](#) (mogwai.properties.properties.Boolean attribute), 24
[has_db_field_prefix](#) (mogwai.properties.properties.DateTime attribute), 25
[has_db_field_prefix](#) (mogwai.properties.properties.DateTimeNaive attribute), 26
[has_db_field_prefix](#) (mogwai.properties.properties.Decimal attribute), 27
[has_db_field_prefix](#) (mogwai.properties.properties.Dictionary attribute), 28
[has_db_field_prefix](#) (mogwai.properties.properties.Double attribute), 30
[has_db_field_prefix](#) (mogwai.properties.properties.Email attribute), 31
[has_db_field_prefix](#) (mogwai.properties.properties.Float attribute), 32
[has_db_field_prefix](#) (mogwai.properties.properties.Integer attribute), 36
[has_db_field_prefix](#) (mogwai.properties.properties.IPV4 attribute), 33
[has_db_field_prefix](#) (mogwai.properties.properties.IPV6 attribute), 34
[has_db_field_prefix](#) (mogwai.properties.properties.IPV6WithV4 attribute), 35
[has_db_field_prefix](#) (mogwai.properties.properties.List attribute), 37
[has_db_field_prefix](#) (mogwai.properties.properties.Long attribute), 39
[has_db_field_prefix](#) (mogwai.properties.properties.PositiveInteger attribute), 40
[has_db_field_prefix](#) (mogwai.properties.properties.PositiveLong attribute), 41
[has_db_field_prefix](#) (mogwai.properties.properties.Short attribute), 42
[has_db_field_prefix](#) (mogwai.properties.properties.Slug attribute), 43
[has_db_field_prefix](#) (mogwai.properties.properties.String attribute), 44
[has_db_field_prefix](#) (mogwai.properties.properties.URL attribute), 45
[has_db_field_prefix](#) (mogwai.properties.properties.UUID attribute), 47
[has_default](#) (mogwai.properties.base.GraphProperty attribute), 23
[has_default](#) (mogwai.properties.properties.Boolean attribute), 24

- has_default (mogwai.properties.properties.DateTime attribute), 25
- has_default (mogwai.properties.properties.DateTimeNaive attribute), 26
- has_default (mogwai.properties.properties.Decimal attribute), 27
- has_default (mogwai.properties.properties.Dictionary attribute), 29
- has_default (mogwai.properties.properties.Double attribute), 30
- has_default (mogwai.properties.properties.Email attribute), 31
- has_default (mogwai.properties.properties.Float attribute), 32
- has_default (mogwai.properties.properties.Integer attribute), 36
- has_default (mogwai.properties.properties.IPV4 attribute), 33
- has_default (mogwai.properties.properties.IPV6 attribute), 34
- has_default (mogwai.properties.properties.IPV6WithV4 attribute), 35
- has_default (mogwai.properties.properties.List attribute), 37
- has_default (mogwai.properties.properties.Long attribute), 39
- has_default (mogwai.properties.properties.PositiveInteger attribute), 40
- has_default (mogwai.properties.properties.PositiveLong attribute), 41
- has_default (mogwai.properties.properties.Short attribute), 42
- has_default (mogwai.properties.properties.Slug attribute), 43
- has_default (mogwai.properties.properties.String attribute), 44
- has_default (mogwai.properties.properties.URL attribute), 45
- has_default (mogwai.properties.properties.UUID attribute), 47
- hist_calls() (mogwai.metrics.manager.MetricManager method), 52
- histogram() (mogwai.metrics.base.MetricsRegistry method), 53
- histogram() (mogwai.metrics.base.RegexRegistry method), 54
- histograms() (mogwai.metrics.manager.MetricManager method), 52
- |
- id (mogwai.models.edge.Edge attribute), 17
- id (mogwai.models.vertex.Vertex attribute), 14
- import_list (mogwai.gremlin.groovy.GroovyImport attribute), 21
- import_name (mogwai.tools.ImportStringError attribute), 55
- import_string() (in module mogwai.tools), 56
- import_strings (mogwai.gremlin.groovy.GroovyImport attribute), 21
- ImportDef (mogwai.gremlin.groovy.GroovyImportParser attribute), 21
- ImportDefn (mogwai.gremlin.groovy.GroovyImportParser attribute), 21
- ImportStringError, 55
- ImportVarName (mogwai.gremlin.groovy.GroovyImportParser attribute), 21
- index() (mogwai.gremlin.groovy.GroovyFunction method), 20
- index() (mogwai.gremlin.groovy.GroovyImport method), 21
- inE() (mogwai.models.vertex.Vertex method), 14
- instance_counter (mogwai.properties.base.GraphProperty attribute), 23
- instance_counter (mogwai.properties.properties.Boolean attribute), 24
- instance_counter (mogwai.properties.properties.DateTime attribute), 25
- instance_counter (mogwai.properties.properties.DateTimeNaive attribute), 26
- instance_counter (mogwai.properties.properties.Decimal attribute), 27
- instance_counter (mogwai.properties.properties.Dictionary attribute), 29
- instance_counter (mogwai.properties.properties.Double attribute), 30
- instance_counter (mogwai.properties.properties.Email attribute), 31
- instance_counter (mogwai.properties.properties.Float attribute), 32
- instance_counter (mogwai.properties.properties.Integer attribute), 36
- instance_counter (mogwai.properties.properties.IPV4 attribute), 33
- instance_counter (mogwai.properties.properties.IPV6 attribute), 34
- instance_counter (mogwai.properties.properties.IPV6WithV4 attribute), 35
- instance_counter (mogwai.properties.properties.List attribute), 38
- instance_counter (mogwai.properties.properties.Long attribute), 39
- instance_counter (mogwai.properties.properties.PositiveInteger attribute), 40

instance_counter (mogwai.properties.properties.PositiveLong attribute), 41

instance_counter (mogwai.properties.properties.Short attribute), 42

instance_counter (mogwai.properties.properties.Slug attribute), 43

instance_counter (mogwai.properties.properties.String attribute), 44

instance_counter (mogwai.properties.properties.URL attribute), 46

instance_counter (mogwai.properties.properties.UUID attribute), 47

Integer (class in mogwai.properties.properties), 36

IntegerValidator (class in mogwai.properties.validators), 50

inV() (mogwai.models.edge.Edge method), 17

inV() (mogwai.models.vertex.Vertex method), 14

IPV4 (class in mogwai.properties.properties), 32

IPV6 (class in mogwai.properties.properties), 33

IPV6WithV4 (class in mogwai.properties.properties), 35

items() (mogwai.gremlin.table.Row method), 22

items() (mogwai.models.edge.Edge method), 18

items() (mogwai.models.vertex.Vertex method), 15

iteritems() (mogwai.gremlin.table.Row method), 22

K

keys() (mogwai.gremlin.table.Row method), 22

keys() (mogwai.models.edge.Edge method), 18

keys() (mogwai.models.vertex.Vertex method), 15

KeywordDef (mogwai.gremlin.groovy.GroovyFunctionParser attribute), 20

klass (mogwai.tools.LazyImportClass attribute), 55

L

label (mogwai.models.edge.Edge attribute), 18

LazyImportClass (class in mogwai.tools), 55

List (class in mogwai.properties.properties), 37

ListValidator (class in mogwai.properties.validators), 50

Long (class in mogwai.properties.properties), 38

LongValidator (class in mogwai.properties.validators), 50

M

message (mogwai.metrics.base.MogwaiMetricsException attribute), 54

message (mogwai.metrics.manager.MogwaiMetricsException attribute), 52

message (mogwai.models.edge.Edge.DoesNotExist attribute), 16

message (mogwai.models.edge.Edge.MultipleObjectsReturned attribute), 16

message (mogwai.models.edge.Edge.WrongElementType attribute), 16

message (mogwai.models.vertex.Vertex.DoesNotExist attribute), 12

message (mogwai.models.vertex.Vertex.MultipleObjectsReturned attribute), 12

message (mogwai.models.vertex.Vertex.WrongElementType attribute), 12

message (mogwai.properties.validators.BaseValidator attribute), 49

message (mogwai.properties.validators.BooleanValidator attribute), 49

message (mogwai.properties.validators.DateTimeUTCValidator attribute), 50

message (mogwai.properties.validators.DateTimeValidator attribute), 50

message (mogwai.properties.validators.DecimalValidator attribute), 50

message (mogwai.properties.validators.DictValidator attribute), 50

message (mogwai.properties.validators.EmailValidator attribute), 50

message (mogwai.properties.validators.FloatValidator attribute), 50

message (mogwai.properties.validators.IntegerValidator attribute), 50

message (mogwai.properties.validators.ListValidator attribute), 50

message (mogwai.properties.validators.LongValidator attribute), 51

message (mogwai.properties.validators.NumericValidator attribute), 51

message (mogwai.properties.validators.PositiveIntegerValidator attribute), 51

message (mogwai.properties.validators.RegexValidator attribute), 51

message (mogwai.properties.validators.StringValidator attribute), 51

message (mogwai.properties.validators.URLValidator attribute), 51

meter() (mogwai.metrics.base.MetricsRegistry method), 53

meter() (mogwai.metrics.base.RegexRegistry method), 54

meter_calls() (mogwai.metrics.manager.MetricManager method), 52

meters() (mogwai.metrics.manager.MetricManager method), 52

MetricManager (class in mogwai.metrics.manager), 51

MetricsRegistry (class in mogwai.metrics.base), 52

mogwai.connection (module), 11, 56

mogwai.gremlin.base (module), 18

mogwai.gremlin.groovy (module), 20

mogwai.gremlin.table (module), 21

mogwai.metrics.base (module), 52

mogwai.metrics.manager (module), 51

mogwai.models.edge (module), 16
 mogwai.models.vertex (module), 12
 mogwai.properties.base (module), 22
 mogwai.properties.properties (module), 24
 mogwai.properties.strategy (module), 48
 mogwai.properties.validators (module), 49
 mogwai.relationships.base (module), 47
 mogwai.spec (module), 56
 mogwai.tools (module), 55
 MogwaiMetricsException, 52, 54
 mro() (mogwai.models.edge.EdgeMetaClass method), 18
 mro() (mogwai.models.vertex.EnumVertexBaseMeta method), 12
 mro() (mogwai.models.vertex.VertexMetaClass method), 16

N

name (mogwai.gremlin.groovy.GroovyFunction attribute), 20
 next() (mogwai.gremlin.table.Row method), 22
 next() (mogwai.gremlin.table.Table method), 21
 NumericValidator (class in mogwai.properties.validators), 51

O

OptionalSpace (mogwai.gremlin.groovy.GroovyImportParser attribute), 21
 outE() (mogwai.models.vertex.Vertex method), 15
 outV() (mogwai.models.edge.Edge method), 18
 outV() (mogwai.models.vertex.Vertex method), 15

P

parse() (in module mogwai.gremlin.groovy), 21
 parse() (mogwai.gremlin.groovy.GroovyFunctionParser class method), 20
 parse() (mogwai.gremlin.groovy.GroovyImportParser class method), 21
 PartitionGraph (class in mogwai.tools), 55
 pop_execute_query_kwargs() (in module mogwai.connection), 11, 57
 PositiveInteger (class in mogwai.properties.properties), 39
 PositiveIntegerValidator (class in mogwai.properties.validators), 51
 PositiveLong (class in mogwai.properties.properties), 40
 pre_save() (mogwai.models.edge.Edge method), 18
 pre_save() (mogwai.models.vertex.Vertex method), 15
 pre_update() (mogwai.models.edge.Edge method), 18
 pre_update() (mogwai.models.vertex.Vertex method), 15
 previous_value (mogwai.properties.base.BaseValueManager attribute), 22

Q

query() (mogwai.models.vertex.Vertex method), 15

query() (mogwai.relationships.base.Relationship method), 48

R

regex (mogwai.properties.validators.EmailValidator attribute), 50
 regex (mogwai.properties.validators.RegexValidator attribute), 51
 regex (mogwai.properties.validators.URLValidator attribute), 51
 RegexRegistry (class in mogwai.metrics.base), 53
 RegexValidator (class in mogwai.properties.validators), 51
 Relationship (class in mogwai.relationships.base), 47
 reload() (mogwai.models.edge.Edge method), 18
 reload() (mogwai.models.vertex.Vertex method), 15
 requires_vertex() (in module mogwai.relationships.base), 48
 Row (class in mogwai.gremlin.table), 21

S

save() (mogwai.models.edge.Edge method), 18
 save() (mogwai.models.vertex.Vertex method), 15
 SaveAlways (class in mogwai.properties.strategy), 48
 SaveOnce (class in mogwai.properties.strategy), 49
 SaveOnChange (class in mogwai.properties.strategy), 48
 SaveOnDecrease (class in mogwai.properties.strategy), 49
 SaveOnIncrease (class in mogwai.properties.strategy), 49
 send_metrics() (mogwai.metrics.base.BaseMetricsReporter method), 54
 SessionPoolManager (class in mogwai.tools), 55
 set_db_field_prefix() (mogwai.properties.base.GraphProperty method), 23
 set_db_field_prefix() (mogwai.properties.properties.Boolean method), 24
 set_db_field_prefix() (mogwai.properties.properties.DateTime method), 25
 set_db_field_prefix() (mogwai.properties.properties.DateTimeNaive method), 26
 set_db_field_prefix() (mogwai.properties.properties.Decimal method), 27
 set_db_field_prefix() (mogwai.properties.properties.Dictionary method), 29
 set_db_field_prefix() (mogwai.properties.properties.Double method), 30

`set_db_field_prefix()` (mogwai.properties.properties.Email method), 31
`set_db_field_prefix()` (mogwai.properties.properties.Float method), 32
`set_db_field_prefix()` (mogwai.properties.properties.Integer method), 36
`set_db_field_prefix()` (mogwai.properties.properties.IPV4 method), 33
`set_db_field_prefix()` (mogwai.properties.properties.IPV6 method), 34
`set_db_field_prefix()` (mogwai.properties.properties.IPV6WithV4 method), 35
`set_db_field_prefix()` (mogwai.properties.properties.List method), 38
`set_db_field_prefix()` (mogwai.properties.properties.Long method), 39
`set_db_field_prefix()` (mogwai.properties.properties.PositiveInteger method), 40
`set_db_field_prefix()` (mogwai.properties.properties.PositiveLong method), 41
`set_db_field_prefix()` (mogwai.properties.properties.Short method), 42
`set_db_field_prefix()` (mogwai.properties.properties.Slug method), 43
`set_db_field_prefix()` (mogwai.properties.properties.String method), 44
`set_db_field_prefix()` (mogwai.properties.properties.URL method), 46
`set_db_field_prefix()` (mogwai.properties.properties.UUID method), 47
`set_property_name()` (mogwai.properties.base.GraphProperty method), 23
`set_property_name()` (mogwai.properties.properties.Boolean method), 24
`set_property_name()` (mogwai.properties.properties.DateTime method), 25
`set_property_name()` (mogwai.properties.properties.DateTimeNaive method), 26
`set_property_name()` (mogwai.properties.properties.Decimal method), 28
`set_property_name()` (mogwai.properties.properties.Dictionary method), 29
`set_property_name()` (mogwai.properties.properties.Double method), 30
`set_property_name()` (mogwai.properties.properties.Email method), 31
`set_property_name()` (mogwai.properties.properties.Float method), 32
`set_property_name()` (mogwai.properties.properties.Integer method), 36
`set_property_name()` (mogwai.properties.properties.IPV4 method), 33
`set_property_name()` (mogwai.properties.properties.IPV6 method), 34
`set_property_name()` (mogwai.properties.properties.IPV6WithV4 method), 35
`set_property_name()` (mogwai.properties.properties.List method), 38
`set_property_name()` (mogwai.properties.properties.Long method), 39
`set_property_name()` (mogwai.properties.properties.PositiveInteger method), 40
`set_property_name()` (mogwai.properties.properties.PositiveLong method), 41
`set_property_name()` (mogwai.properties.properties.Short method), 42
`set_property_name()` (mogwai.properties.properties.PositiveLong method), 41
`set_property_name()` (mogwai.properties.properties.Short method), 42
`set_property_name()` (mogwai.properties.properties.Slug method), 43
`set_property_name()` (mogwai.properties.properties.String method), 45
`set_property_name()` (mogwai.properties.properties.URL method), 46
`set_property_name()` (mogwai.properties.properties.UUID method), 47
`setup()` (in module mogwai.connection), 11, 57
`setup_registry()` (mogwai.metrics.base.BaseMetricsReporter method), 54
`setup_reporters()` (mogwai.metrics.manager.MetricManager method), 52
`setval()` (mogwai.properties.base.BaseValueManager method), 22
`Short` (class in mogwai.properties.properties), 42
`should_save()` (mogwai.properties.base.GraphProperty method), 23

- should_save() (mogwai.properties.properties.Boolean method), 24
 should_save() (mogwai.properties.properties.DateTime method), 25
 should_save() (mogwai.properties.properties.DateTimeNaive method), 26
 should_save() (mogwai.properties.properties.Decimal method), 28
 should_save() (mogwai.properties.properties.Dictionary method), 29
 should_save() (mogwai.properties.properties.Double method), 30
 should_save() (mogwai.properties.properties.Email method), 31
 should_save() (mogwai.properties.properties.Float method), 32
 should_save() (mogwai.properties.properties.Integer method), 37
 should_save() (mogwai.properties.properties.IPV4 method), 33
 should_save() (mogwai.properties.properties.IPV6 method), 34
 should_save() (mogwai.properties.properties.IPV6WithV4 method), 35
 should_save() (mogwai.properties.properties.List method), 38
 should_save() (mogwai.properties.properties.Long method), 39
 should_save() (mogwai.properties.properties.PositiveInteger method), 40
 should_save() (mogwai.properties.properties.PositiveLong method), 41
 should_save() (mogwai.properties.properties.Short method), 42
 should_save() (mogwai.properties.properties.Slug method), 44
 should_save() (mogwai.properties.properties.String method), 45
 should_save() (mogwai.properties.properties.URL method), 46
 should_save() (mogwai.properties.properties.UUID method), 47
 Slug (class in mogwai.properties.properties), 43
 start() (mogwai.metrics.base.BaseMetricsReporter method), 54
 start() (mogwai.metrics.manager.MetricManager method), 52
 start_reporter() (mogwai.metrics.base.BaseMetricsReporter method), 54
 stop() (mogwai.metrics.base.BaseMetricsReporter method), 54
 stop() (mogwai.metrics.manager.MetricManager method), 52
 Strategy (class in mogwai.properties.strategy), 49
 String (class in mogwai.properties.properties), 44
 StringValidator (class in mogwai.properties.validators), 51
 sync_spec() (in module mogwai.connection), 12, 57
T
 Table (class in mogwai.gremlin.table), 21
 Text (in module mogwai.properties.properties), 45
 time_calls() (mogwai.metrics.manager.MetricManager method), 52
 timer() (mogwai.metrics.base.MetricsRegistry method), 53
 timer() (mogwai.metrics.base.RegexRegistry method), 54
 timers() (mogwai.metrics.manager.MetricManager method), 52
 to_database() (mogwai.properties.base.GraphProperty method), 23
 to_database() (mogwai.properties.properties.Boolean method), 24
 to_database() (mogwai.properties.properties.DateTime method), 26
 to_database() (mogwai.properties.properties.DateTimeNaive method), 27
 to_database() (mogwai.properties.properties.Decimal method), 28
 to_database() (mogwai.properties.properties.Dictionary method), 29
 to_database() (mogwai.properties.properties.Double method), 30
 to_database() (mogwai.properties.properties.Email method), 31
 to_database() (mogwai.properties.properties.Float method), 32
 to_database() (mogwai.properties.properties.Integer method), 37
 to_database() (mogwai.properties.properties.IPV4 method), 33
 to_database() (mogwai.properties.properties.IPV6 method), 34
 to_database() (mogwai.properties.properties.IPV6WithV4 method), 35
 to_database() (mogwai.properties.properties.List method), 38
 to_database() (mogwai.properties.properties.Long method), 39
 to_database() (mogwai.properties.properties.PositiveInteger method), 40
 to_database() (mogwai.properties.properties.PositiveLong method), 41
 to_database() (mogwai.properties.properties.Short method), 43
 to_database() (mogwai.properties.properties.Slug method), 44

- `to_database()` (mogwai.properties.properties.String method), 45
`to_database()` (mogwai.properties.properties.URL method), 46
`to_database()` (mogwai.properties.properties.UUID method), 47
`to_python()` (mogwai.properties.base.GraphProperty method), 23
`to_python()` (mogwai.properties.properties.Boolean method), 24
`to_python()` (mogwai.properties.properties.DateTime method), 26
`to_python()` (mogwai.properties.properties.DateTimeNaive method), 27
`to_python()` (mogwai.properties.properties.Decimal method), 28
`to_python()` (mogwai.properties.properties.Dictionary method), 29
`to_python()` (mogwai.properties.properties.Double method), 30
`to_python()` (mogwai.properties.properties.Email method), 31
`to_python()` (mogwai.properties.properties.Float method), 32
`to_python()` (mogwai.properties.properties.Integer method), 37
`to_python()` (mogwai.properties.properties.IPV4 method), 33
`to_python()` (mogwai.properties.properties.IPV6 method), 34
`to_python()` (mogwai.properties.properties.IPV6WithV4 method), 36
`to_python()` (mogwai.properties.properties.List method), 38
`to_python()` (mogwai.properties.properties.Long method), 39
`to_python()` (mogwai.properties.properties.PositiveInteger method), 40
`to_python()` (mogwai.properties.properties.PositiveLong method), 41
`to_python()` (mogwai.properties.properties.Short method), 43
`to_python()` (mogwai.properties.properties.Slug method), 44
`to_python()` (mogwai.properties.properties.String method), 45
`to_python()` (mogwai.properties.properties.URL method), 46
`to_python()` (mogwai.properties.properties.UUID method), 47
`transform_params_to_database()` (mogwai.gremlin.base.BaseGremlinMethod method), 19
`transform_params_to_database()` (mogwai.gremlin.base.GremlinMethod method), 19
`transform_params_to_database()` (mogwai.gremlin.base.GremlinTable method), 19
`transform_params_to_database()` (mogwai.gremlin.base.GremlinValue method), 20
`translate_db_fields()` (mogwai.models.edge.Edge method), 18
`translate_db_fields()` (mogwai.models.vertex.Vertex method), 15
- ## U
- `update()` (mogwai.models.edge.Edge method), 18
`update()` (mogwai.models.vertex.Vertex method), 15
URL (class in mogwai.properties.properties), 45
URLValidator (class in mogwai.properties.validators), 51
UUID (class in mogwai.properties.properties), 46
- ## V
- `validate()` (mogwai.models.edge.Edge method), 18
`validate()` (mogwai.models.vertex.Vertex method), 15
`validate()` (mogwai.properties.base.GraphProperty method), 23
`validate()` (mogwai.properties.properties.Boolean method), 25
`validate()` (mogwai.properties.properties.DateTime method), 26
`validate()` (mogwai.properties.properties.DateTimeNaive method), 27
`validate()` (mogwai.properties.properties.Decimal method), 28
`validate()` (mogwai.properties.properties.Dictionary method), 29
`validate()` (mogwai.properties.properties.Double method), 30
`validate()` (mogwai.properties.properties.Email method), 31
`validate()` (mogwai.properties.properties.Float method), 32
`validate()` (mogwai.properties.properties.Integer method), 37
`validate()` (mogwai.properties.properties.IPV4 method), 33
`validate()` (mogwai.properties.properties.IPV6 method), 34
`validate()` (mogwai.properties.properties.IPV6WithV4 method), 36
`validate()` (mogwai.properties.properties.List method), 38
`validate()` (mogwai.properties.properties.Long method), 39

- validate() (mogwai.properties.properties.PositiveInteger method), 40
- validate() (mogwai.properties.properties.PositiveLong method), 41
- validate() (mogwai.properties.properties.Short method), 43
- validate() (mogwai.properties.properties.Slug method), 44
- validate() (mogwai.properties.properties.String method), 45
- validate() (mogwai.properties.properties.URL method), 46
- validate() (mogwai.properties.properties.UUID method), 47
- validate_field() (mogwai.models.edge.Edge method), 18
- validate_field() (mogwai.models.vertex.Vertex method), 15
- validator (mogwai.properties.base.GraphProperty attribute), 23
- validator (mogwai.properties.properties.Boolean attribute), 25
- validator (mogwai.properties.properties.DateTime attribute), 26
- validator (mogwai.properties.properties.DateTimeNaive attribute), 27
- validator (mogwai.properties.properties.Decimal attribute), 28
- validator (mogwai.properties.properties.Dictionary attribute), 29
- validator (mogwai.properties.properties.Double attribute), 30
- validator (mogwai.properties.properties.Email attribute), 31
- validator (mogwai.properties.properties.Float attribute), 32
- validator (mogwai.properties.properties.Integer attribute), 37
- validator (mogwai.properties.properties.IPV4 attribute), 33
- validator (mogwai.properties.properties.IPV6 attribute), 35
- validator (mogwai.properties.properties.IPV6WithV4 attribute), 36
- validator (mogwai.properties.properties.List attribute), 38
- validator (mogwai.properties.properties.Long attribute), 39
- validator (mogwai.properties.properties.PositiveInteger attribute), 40
- validator (mogwai.properties.properties.PositiveLong attribute), 42
- validator (mogwai.properties.properties.Short attribute), 43
- validator (mogwai.properties.properties.Slug attribute), 44
- validator (mogwai.properties.properties.String attribute), 45
- validator (mogwai.properties.properties.URL attribute), 46
- validator (mogwai.properties.properties.UUID attribute), 47
- value_manager (mogwai.properties.base.GraphProperty attribute), 23
- value_manager (mogwai.properties.properties.Boolean attribute), 25
- value_manager (mogwai.properties.properties.DateTime attribute), 26
- value_manager (mogwai.properties.properties.DateTimeNaive attribute), 27
- value_manager (mogwai.properties.properties.Decimal attribute), 28
- value_manager (mogwai.properties.properties.Dictionary attribute), 29
- value_manager (mogwai.properties.properties.Double attribute), 30
- value_manager (mogwai.properties.properties.Email attribute), 31
- value_manager (mogwai.properties.properties.Float attribute), 32
- value_manager (mogwai.properties.properties.Integer attribute), 37
- value_manager (mogwai.properties.properties.IPV4 attribute), 33
- value_manager (mogwai.properties.properties.IPV6 attribute), 35
- value_manager (mogwai.properties.properties.IPV6WithV4 attribute), 36
- value_manager (mogwai.properties.properties.List attribute), 38
- value_manager (mogwai.properties.properties.Long attribute), 39
- value_manager (mogwai.properties.properties.PositiveInteger attribute), 40
- value_manager (mogwai.properties.properties.PositiveLong attribute), 42
- value_manager (mogwai.properties.properties.Short attribute), 43
- value_manager (mogwai.properties.properties.Slug attribute), 44
- value_manager (mogwai.properties.properties.String attribute), 45
- value_manager (mogwai.properties.properties.URL attribute), 46
- value_manager (mogwai.properties.properties.UUID attribute), 47
- values() (mogwai.gremlin.table.Row method), 22
- values() (mogwai.models.edge.Edge method), 18
- values() (mogwai.models.vertex.Vertex method), 16

VarName (mogwai.gremlin.groovy.GroovyFunctionParser attribute), 20
Vertex (class in mogwai.models.vertex), 12
Vertex.DoesNotExist, 12
Vertex.MultipleObjectsReturned, 12
Vertex.WrongElementType, 12
VertexMetaClass (class in mogwai.models.vertex), 16
vertices() (mogwai.relationships.base.Relationship method), 48

W

write_compiled_indices_to_file() (in module mogwai.spec), 56
write_diff_indices_to_file() (in module mogwai.spec), 56
write_specs_to_file() (in module mogwai.spec), 56