
Ojota Documentation

Release 3.0.0

Felipe Lerena, Juan Pedro Fisanoti, Ezequiel Chan, Nicolas Sarub

August 22, 2017

1	How does it works?	3
2	Supported data formats	5
3	New Features for 2.0	7
4	Installation	9
5	Optional dependencies	11
6	Table of contents	13
6.1	Modules Reference	13
6.2	Examples	16
7	Indices and tables	21

Ojota is a ORM and flat file database engine.

Ojota is Free Software! you can check the code at <https://github.com/MSA-Argentina/ojota>

How does it works?

First we will define the data object

```
# The information is stored in a file called Persons.json
class Person(Ojota):
    required_fields = ("name", "address", "age")
    cache = Memcache()

# The information is stored in a file called Teams.yaml
class Team(Ojota):
    pk_field = "id"
    data_source = YAMLSource()
    required_fields = ("id", "name", "color")

    def __repr__(self):
        return self.name
```

Just with that we can query the ORM objects

```
# Some Example queries
# "all" returns all the Person Objects
Person.all()
# "many" will return filtered results
Person.many(age=30, sorted="name")
Person.many(age__lt=30, sorted="-name")
Person.many(sorted="name")

# "one" will get only one object
Team.one(1) # you can just send the primary key
Team.one(name="River Plate")

# You can sub-query over the results
persons = Person.all()
elders = persons.many(age__gt=30)
fat_elders = elders.many(weight__gt=50)
female_elders = elders.many(gender="F")
```

That's it your information will be stored in plain text and you will have a powerfull ORM to play with it

Supported data formats

- JSON
- DSON
- YAML
- CSV
- JSON through web service
- XLS

New Features for 2.0

- QuerySets with recursive filtering
- “Callbacks” support (you can add custom properties with a callback function)
- Hierarchical Objects support

Installation

With easy_install

```
sudo easy_install ojota
```

With pip

```
sudo pip install ojota
```

From source

```
hg clone ssh://hg@bitbucket.org/msa_team/ojota
sudo python setup.py install
```

Optional dependencies

- pyyaml - To fetch the data from a file with YAML format
- dogeon - To fetch the data from a file with DSON format
- request - To fetch JSON from web service
- flask – To run the example web service.

You might also want to take a look at Ojota's sister project called Havaiana <http://havaiana.rtfid.org>

Table of contents

Modules Reference

base

class `base.Relation` (*attr_fk, to_class, related_name=None*)

Adds a relation to another object.

get_property ()

Returns the property in which the relation will be referenced.

set_reversed_property (*from_class*)

Returns the property in which the backwards relation will be referenced.

class `base.MetaOjota` (**args, **kwargs*)

Metaclass for Ojota

class `base.Ojota` (*_pk=None, **kwargs*)

Base class to create instances of serialized data in the source files.

__eq__ (*other*)

Compare the equality of two elements.

__init__ (*_pk=None, **kwargs*)

Constructor.

__repr__ ()

String representation of the elements.

__weakref__

list of weak references to the object (if defined)

classmethod `_filter` (*data, filters*)

Applies filter to data.

Arguments: data – an iterable containing the data filters – a dictionary with the filters

classmethod `_objetrize` (*data*)

Return the data into an element.

classmethod `_read_all_from_datasource` ()

Reads the data from the datasource, makes a dictionary with the key specified in the key parameter. Allows to filter by subdirectories when the data is not on the root according to the data path.

classmethod `_read_item_from_datasource` (*pk*)

Reads the data form the datasource if support index search.

classmethod `_sort` (*data_list*, *order_fields*)

Sort a list by a given field or field froups.

Arguments: *data_list* – a list with the data *order_fields* – a string with the order fields

classmethod `_test_expression` (*expression*, *value*, *element_data*)

Finds out if a value in a given field matches an expression.

Arguments: *expression* – a string with the comparison expression. If the expression is a field name it will be compared with equal. In case that the field has “__” and an operation appended the it is compared with the appended expression. The available expressions allowed are: “=”, “exact”, “iexact”, “contains”, “icontains”, “in”, “gt”, “gte”, “lt”, “lte”, “startswith”, “istartswith”, “endswith”, “iendswith”, “range” and “ne”

dump_values (*new_data=None*, *delete=False*)

Saves the data into a file.

classmethod `many` (***kargs*)

Returns all the elements that match the conditions.

classmethod `one` (*pk=None*, ***kargs*)

Returns the first element that matches the conditions.

primary_key

Returns the primary key value.

save ()

Save function for an object.

update (***kwargs*)

Updates the given values.

sources

class `sources.Source` (*data_path=None*, *create_empty=True*)

Base class for all the data sources.

`__init__` (*data_path=None*, *create_empty=True*)

Constructor for the Source class.

Arguments: *data_path* – the path where the data is located.

`__weakref__`

list of weak references to the object (if defined)

`_get_file_path` (*cls*)

Builds the path where the data will be located.

Arguments: *cls* – the class with the data.

fetch_element (*cls*, *pk*)

Fetch the elements for a given element of a class.

Arguments: *cls* - the class with the data. *pk* - the primary key of the given element.

fetch_elements (*cls*)

Fetch the elements for a given class.

Arguments: *cls* - the class with the data.

save (*cls*, *data*)

Fetch the elements for a given element of a class.

Arguments: *cls* - the class with the data. *pk* - the primary key of the given element.

```
class sources.JSONSource (data_path=None, create_empty=True, indent=4)
    Source class for the data stored with JSON format

    __init__ (data_path=None, create_empty=True, indent=4)
        Constructor for the Source class.
        Arguments: data_path – the path where the data is located. create_empty – if file in data_path
            is not found, create an empty one. indent – control the indentation of the JSON in the file.

    read_elements (cls, filepath)
        Reads the elements form a JSON file. Returns a dictionary containing the read data.
        Arguments: filepath – the path for the json file.

class sources.YAMLSource (data_path=None, create_empty=True)
    Source class for the data stored with YAML format.

    requires the PyYaml package to run.

    read_elements (cls, filepath)
        Reads the elements form a JSON file. Returns a dictionary containing the read data.
        Arguments: filepath – the path for the json file.

class sources.WebServiceSource (data_path=None, method='get', get_all_cmd='/all',
                                get_cmd='/data', user=None, password=None,
                                cert=None, custom_call=None)
    Source class for the data stored with JSON format taken through a Web Service.

    Requires the “requests” package to run. http://pypi.python.org/pypi/requests

    __init__ (data_path=None, method='get', get_all_cmd='/all', get_cmd='/data',
              user=None, password=None, cert=None, custom_call=None)
        Constructor for the WebServiceSource class.
        Arguments: data_path – the path where the data is located. method – the http method that will
            be used with the web service. Defaults to “get”. get_all_cmd – the WS command to fetch
            all the data. Defaults to “/all”. get_cmd – the WS command to fetch one element. Defaults
            to “/data” user – the user name for the authentication. If not provided the request will not
            use authentication. password – the password for the authentication. If not provided the
            request will not use authentication.

    read_element (cls, url, pk)
        Reads one element elements form a JSON file. Returns a dictionary containing the read data.
        Arguments: cls – the data class. url – the path for the WS. pk – the primary key.

    read_elements (cls, url)
        Reads the elements form a WS request. Returns a dictionary containing the read data.
        Arguments: cls – the data class. url – the path for the WS.
```

cache

```
class cache.Cache
    The base Cache class. Stores the cached data in memory.

    __contains__ (name)
        Returns True if a given element is cached.
        Arguments: name – the cache name.

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

    get (name)
        Gets the data from cache.
```

Arguments: name – the cache name.

set (name, elems)

Sets the data into cache.

Arguments: name – the cache name. elems – the data to cache.

class cache.**Memcache** (cache_location='127.0.0.1', port=11211, expiration_time=None, debug=None)

Stores the cached data in memcache.

__contains__ (name)

Returns True if a given element is cached.

Arguments: name – the cache name.

__init__ (cache_location='127.0.0.1', port=11211, expiration_time=None, debug=None)

Constructor for the Memcache class.

Arguments: cache_location – memcached URI. defaults to 127.0.0.1 port – memcached port.

Defaults to 11211 expiration_time – memcache expiration time debug – activate memcache debug. Defaults to None

get (name)

Gets the data from cache.

Arguments: name – the cache name.

set (name, elems)

Sets the data into cache.

Arguments: name – the cache name. elems – the data to cache.

class cache.**DummyCache**

Dummy Cache class to be able to use no cache.

set (name, elems)

Sets the data into cache.

Arguments: name – the cache name. elems – the data to cache.

Examples

Examples for data stored locally

```
"""
This file is part of Ojota.

Ojota is free software: you can redistribute it and/or modify
it under the terms of the GNU LESSER GENERAL PUBLIC LICENSE as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Ojota is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License
along with Ojota. If not, see <http://www.gnu.org/licenses/>.
"""
from __future__ import absolute_import
import os
```

```

from ojota import Ojota, Relation, set_data_source, Callback
from ojota.base import OjotaHierarchy
from ojota.cache import Memcache
from ojota.examples.example_ws import Country, Flag
from ojota.sources import YAMLSource, CSVSource, XLSSource

file_path = (os.path.dirname(os.path.abspath(__file__)))
set_data_source(os.path.join(file_path, "data"))

class Team(Ojota):
    plural_name = "Teams"
    pk_field = "id"
    data_source = YAMLSource()
    required_fields = ("id", "name", "color")

    def __repr__(self):
        return self.name

class Person(Ojota):
    plural_name = "Persons"
    pk_field = "id"
    required_fields = ("id", "name", "address", "age", "team_id")
    team = Relation("team_id", Team, "persons")
    country = Relation("country_id", Country, "persons")
    age_str = Callback("age", lambda x:str(x))

    def __repr__(self):
        return self.name

class Customer(Ojota):
    plural_name = "Customers"
    pk_field = "id"
    required_fields = ("id", "name", "address", "age")
    data_source = CSVSource()

class OtherPeople(Ojota):
    plural_name = "OtherPeople"
    pk_field = "id"
    data_source = XLSSource()
    required_fields = ("id", "name", "last_name", "age")

class Place(OjotaHierarchy):
    plural_name = "Places"
    pk_field = "id"
    required_fields = ("id", "name")
    default_order = ("id")

```

Examples for data taken through Web Service

- includes web service example

```
"""
This file is part of Ojota.

Ojota is free software: you can redistribute it and/or modify
it under the terms of the GNU LESSER GENERAL PUBLIC LICENSE as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Ojota is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License
along with Ojota. If not, see <http://www.gnu.org/licenses/>.
"""
from __future__ import absolute_import
try:
    from flask import Flask
except:
    pass

from json import dumps

from ojota import Ojota, Relation
from ojota.sources import WebServiceSource

class Flag(Ojota):
    plural_name = "Flags"
    pk_field = "id"
    required_fields = ("id", "description")
    data_source = WebServiceSource("http://localhost:8001")

    def __repr__(self):
        return self.description

class Country(Ojota):
    plural_name = "Countries"
    pk_field = "id"
    required_fields = ("id", "name")
    data_source = WebServiceSource("http://localhost:8001")
    country = Relation("flag_id", Flag, "countries")

    def __repr__(self):
        return self.name

if __name__ == "__main__":
    app = Flask(__name__)

    @app.route("/Countries/all")
    @app.route("/Countries/<id_>/data")
    def all_countries(id_=None):
        data = [{"id": "0", "name": "Argentina", "flag_id": "0"},
                {"id": "1", "name": "Brazil", "flag_id": "1"}]
        if id_ is None:
```

```
        ret = dumps(data)
    else:
        ret = dumps(data[int(id_)])
    return ret

@app.route("/Flags/all")
@app.route("/Flags/<id_>/data")
def all_flags(id_=None):
    data = [{"id": "0", "description": "Blue and White"},
            {"id": "1", "description": "Green, Yellow and Blue"}]
    if id_ is None:
        ret = dumps(data)
    else:
        ret = dumps(data[int(id_)])
    return ret
app.debug = True
app.run(port=8001)
```

Indices and tables

- *genindex*
- *modindex*
- *search*

Symbols

[__contains__\(\) \(cache.Cache method\), 15](#)
[__contains__\(\) \(cache.Memcache method\), 16](#)
[__eq__\(\) \(base.Ojota method\), 13](#)
[__init__\(\) \(base.Ojota method\), 13](#)
[__init__\(\) \(cache.Memcache method\), 16](#)
[__init__\(\) \(sources.JSONSource method\), 15](#)
[__init__\(\) \(sources.Source method\), 14](#)
[__init__\(\) \(sources.WebServiceSource method\), 15](#)
[__repr__\(\) \(base.Ojota method\), 13](#)
[__weakref__ \(base.Ojota attribute\), 13](#)
[__weakref__ \(cache.Cache attribute\), 15](#)
[__weakref__ \(sources.Source attribute\), 14](#)
[_filter\(\) \(base.Ojota class method\), 13](#)
[_get_file_path\(\) \(sources.Source method\), 14](#)
[_objetrize\(\) \(base.Ojota class method\), 13](#)
[_read_all_from_datasource\(\) \(base.Ojota class method\), 13](#)
[_read_item_from_datasource\(\) \(base.Ojota class method\), 13](#)
[_sort\(\) \(base.Ojota class method\), 14](#)
[_test_expression\(\) \(base.Ojota class method\), 14](#)

C

[Cache \(class in cache\), 15](#)

D

[DummyCache \(class in cache\), 16](#)
[dump_values\(\) \(base.Ojota method\), 14](#)

F

[fetch_element\(\) \(sources.Source method\), 14](#)
[fetch_elements\(\) \(sources.Source method\), 14](#)

G

[get\(\) \(cache.Cache method\), 15](#)
[get\(\) \(cache.Memcache method\), 16](#)
[get_property\(\) \(base.Relation method\), 13](#)

J

[JSONSource \(class in sources\), 14](#)

M

[many\(\) \(base.Ojota class method\), 14](#)
[Memcache \(class in cache\), 16](#)
[MetaOjota \(class in base\), 13](#)

O

[Ojota \(class in base\), 13](#)
[one\(\) \(base.Ojota class method\), 14](#)

P

[primary_key \(base.Ojota attribute\), 14](#)

R

[read_element\(\) \(sources.WebServiceSource method\), 15](#)
[read_elements\(\) \(sources.JSONSource method\), 15](#)
[read_elements\(\) \(sources.WebServiceSource method\), 15](#)
[read_elements\(\) \(sources.YAMLSource method\), 15](#)
[Relation \(class in base\), 13](#)

S

[save\(\) \(base.Ojota method\), 14](#)
[save\(\) \(sources.Source method\), 14](#)
[set\(\) \(cache.Cache method\), 16](#)
[set\(\) \(cache.DummyCache method\), 16](#)
[set\(\) \(cache.Memcache method\), 16](#)
[set_reversed_property\(\) \(base.Relation method\), 13](#)
[Source \(class in sources\), 14](#)

U

[update\(\) \(base.Ojota method\), 14](#)

W

[WebServiceSource \(class in sources\), 15](#)

Y

[YAMLSource \(class in sources\), 15](#)