Django redis admin Documentation

Release 0.3.0

Rick van Hattem (Wolph)

Aug 21, 2023

Contents

Source: https://github.com/WoLpH/django-redis-admin

Contents:

CHAPTER 1

Django Redis Admin

1.1 Introduction

With *django-redis-admin* you can view (and in the future, edit) your Redis databases. It supports simple servers, master slave setups and sentinel setups.

The admin works by creating a *RedisQueryset* which fakes Django models and querysets so the *ModelAdmin* thinks it's using a regular database backed model.

Since Redis only supports basic types the library allows for optional *base64* encoding/decoding and *json* encoding/decoding.

While I would not recommend using it as a regular queryset to access Redis. In addition to querying data it does some extra queries which you usually don't need (such as fetching idle data) and it does some automatic conversion steps.

1.2 Requirements

- Python *3.6* and above
- Django (tested with 2.1, probably works with any version that supports Python 3)
- Python-redis (*pip install redis*)

1.3 Installation

django-redis-admin can be installed via pip.

```
pip install django-redis-admin
```

Then just add *redis_admin* to your *INSTALLED_APPS*.

Optionally, configure your servers if you have multiple and/or non-standard (i.e. non-localhost) redis servers.

Below are several example configurations. The default settings can always be found in *redis_admin/settings.py*

You can run the demo project using the following commands:

```
cd test_redis_admin
python manage.py runserver
```

The default username/password is *admin*/*admin*: http://localhost:8080/admin/

1.3.1 Basic configuration

```
# https://redis-py.readthedocs.io/en/latest/index.html#redis.Redis
REDIS_SERVERS = dict(
    localhost=dict(),
)
```

1.3.2 Explicit configuration

```
# https://redis-py.readthedocs.io/en/latest/index.html#redis.Redis
REDIS_SERVERS = dict(
    redis_server_a=dict(host='127.0.0.1', port=6379, db=0),
)
```

1.3.3 Master slave configuration

```
# https://redis-py.readthedocs.io/en/latest/index.html#redis.Redis
REDIS_SERVERS = dict(
    redis_server_a=dict(
      master=dict(host='master_hostname', port=6379, db=0),
      slave=dict(host='slave_hostname', port=6379, db=0),
   )
)
```

1.3.4 Sentinel Configuration

```
# The `REDIS_SENTINELS` setting should be a list containing host/port
# combinations. As documented here:
# https://github.com/andymccurdy/redis-py/blob/master/README.rst#sentinel-support
REDIS_SENTINELS = [('server_a', 26379), ('server_b', 26379)]

# The `REDIS_SENTINEL_OPTIONS` are the extra arguments to
# `redis.sentinel.Sentinel`:
# https://github.com/andymccurdy/redis-py/blob/
↪cdfe2befbe00db4a3c48c9ddd6d64dea15f6f0db/redis/sentinel.py#L128-L155
REDIS_SENTINEL_OPTIONS = dict(socket_timeout=0.1)

# The `service_name` is used to find the server within the Sentinel
# configuration. The dictionary key will be used as the name in the admin
# https://redis-py.readthedocs.io/en/latest/index.html#redis.Redis
REDIS_SERVERS = dict(
```

(continues on next page)

(continued from previous page)

```
    name_in_admin=dict(service_name='name_in_sentinel'),
    other_server=dict(service_name='other_server'),
)
```

1.3.5 Base64 and/or JSON decoding

As a convenient option all values can optionally be *base64* and/or *json* encoded. To configure this a regular expression can be specified which will be matched against the keys.

```
# For all keys
REDIS_JSON_KEY_RE = '.*'
REDIS_BASE64_KEY_RE = '.*'

# Keys starting with a pattern:
REDIS_BASE64_KEY_RE = '^some_prefix.*'

# Keys ending with a pattern:
REDIS_JSON_KEY_RE = '.*some_suffix$'
```

And if a specific *json* decoder is needed, the *json* module can be specified. The module needs to be importable and have a *dumps* and *loads* method. By default it simply imports the *json* module:

```
REDIS_JSON_MODULE = 'json'
```

1.3.6 Representation cropping

Within the Django Admin list view the values are cropped by default to prevent really long lines. This size can be adjusted through:

```
REDIS_REPR_CROP_SIZE = 150
```

1.4 TODO

- Allow saving values
- Allow deleting values
- Support Redis Bitmaps
- Support Redis HyperLogLogs

CHAPTER 2

redis_admin package

2.1 Submodules

2.2 redis_admin.settings module

redis_admin.settings.**CROP_SIZE = 150**
The maximum amount of characters to show before cropping them in the admin list view

redis_admin.settings.**JSON_MODULE = <module 'json' from '/home/docs/.pyenv/versions/3.7.9/l**
Can be any importable module that has a *loads* and *dumps* function

redis_admin.settings.**SENTINELS = []**
The *REDIS_SENTINELS* setting should be a list containing host/port combinations. As documented here: https://github.com/andymccurdy/redis-py/blob/master/README.rst#sentinel-support For example: [('server_a', 26379), ('server_b', 26379)]

redis_admin.settings.**SENTINEL_OPTIONS = {'socket_timeout': 0.3}**
The *REDIS_SENTINEL_OPTIONS* are the extra arguments to *redis.sentinel.Sentinel*: https://github.com/andymccurdy/redis-py/blob/cdfe2befbe00db4a3c48c9ddd6d64dea15f6f0db/redis/sentinel.py#L128-L155

redis_admin.settings.**SERVERS = {'default': {}}**
To use a separate *master/slave* configuration *master/slave* sub-dictionaries can be provided. Otherwise the top-level dictionary will be passed along to *redis.Redis*: https://redis-py.readthedocs.io/en/latest/index.html#redis.Redis

redis_admin.settings.**SOCKET_TIMEOUT = 0.3**
Default socket timeout if no other settings are given.

2.3 redis_admin.client module

redis_admin.client.**get_master**(*name*) → redis.client.Redis

redis_admin.client.**get_sentinel**()

redis_admin.client.**get_slave**(*name*) → redis.client.Redis

2.4 redis_admin.models module

class redis_admin.models.**Default**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*)
Bases: *redis_admin.models.RedisValue*

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

objects = <django.db.models.manager.Manager object>

class redis_admin.models.**Meta**
Bases: *redis_admin.models.RedisMeta*

class redis_admin.models.**RedisHash**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*)
Bases: *redis_admin.models.RedisValue*

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

fetch_value(*client: redis.client.Redis*)
Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

objects = <django.db.models.manager.Manager object>

rediszset
Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

value

class redis_admin.models.**RedisList**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*)
Bases: *redis_admin.models.RedisValue*

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

fetch_value(*client: redis.client.Redis*)
Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

objects = <django.db.models.manager.Manager object>

redisset
Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

value

class redis_admin.models.**RedisMeta**

Bases: `object`

get_field(*name*)

managed = False

class redis_admin.models.**RedisSet**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*, *redislist_ptr*)

Bases: *redis_admin.models.RedisList*

exception DoesNotExist

Bases: `redis_admin.models.DoesNotExist`

exception MultipleObjectsReturned

Bases: `redis_admin.models.MultipleObjectsReturned`

fetch_value(*client: redis.client.Redis*)

Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

redislist_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

redislist_ptr_id

value

class redis_admin.models.**RedisString**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*)

Bases: *redis_admin.models.RedisValue*

exception DoesNotExist

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `django.core.exceptions.MultipleObjectsReturned`

fetch_value(*client: redis.client.Redis*)

Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

objects = <django.db.models.manager.Manager object>

value

class redis_admin.models.**RedisValue**(**args*, ***kwargs*)

Bases: `django.db.models.base.Model`

class Meta

Bases: *redis_admin.models.RedisMeta*

abstract = False

TYPES = {'hash': <class 'redis_admin.models.RedisHash'>, 'list': <class 'redis_admin

base64
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classmethod create(*type*, ***kwargs*)

cropped_value

decode_string(*raw_value*)

expires_at
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

fetch_value(*client: redis.client.Redis*)
Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

get_cropped_value(*crop_size*)

idle

idle_since
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

json
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

key
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

raw_value
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

classmethod register_type(*type*)

ttl

type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

value

class redis_admin.models.**RedisZSet**(*key*, *raw_value*, *type*, *expires_at*, *idle_since*, *base64*, *json*, *redishash_ptr*)
Bases: *redis_admin.models.RedisHash*

exception DoesNotExist
Bases: redis_admin.models.DoesNotExist

exception MultipleObjectsReturned
Bases: redis_admin.models.MultipleObjectsReturned

fetch_value(*client: redis.client.Redis*)
Fetch the value. Note that if a pipe is passed as *client* the result will be in the *pipe.execute()* instead

redishash_ptr
Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

redishash_ptr_id

value

redis_admin.models.**decode_bytes**(*value*, *encoding='utf-8'*, *method='replace'*)

2.5 redis_admin.admin module

class redis_admin.admin.**Query**(*queryset*)

Bases: object

order_by = ()

select_related(**args*, ***kwargs*)

class redis_admin.admin.**Queryset**(*model: redis_admin.models.RedisValue*, *slice_limit=101*)

Bases: object

count()

filter(**filters*, ***raw_filters*)

get(**args*, ***kwargs*)

order_by(**args*, ***kwargs*)

class redis_admin.admin.**RedisAdmin**(*model*, *admin_site*)

Bases: django.contrib.admin.options.ModelAdmin

get_queryset(*request*)

Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

list_display = ['key', 'type', 'expires_at', 'ttl', 'idle', 'cropped_value', 'json', '

media

readonly_fields = ['key', 'raw_value', 'type', 'expires_at', 'idle_since', 'base64', '

search_fields = ('key__contains',)

show_full_result_count = False

redis_admin.admin.**grouper**(*iterable*, *n*, *fillvalue=None*)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

Q

R

S

T

V