django-dnsmanager

Jan 18, 2021

Contents

This is a DNS manager Django app.

CHAPTER 1

Installation

The following lines creates a Python3 virtualenv and installs `django-dnsmanager` inside.

```
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install django-dnsmanager
```

CHAPTER 2

Features

- Polymorphic models based on Django Polymorphic ;
- Integration with Django Contrib Admin and AdminDocs ;
- Integration with Django Rest Framework ;
- Generation of ready to use zone files.

This app targets Django 2.2 (last LTS and current Debian version (from Debian 11 Bullseye)) and 3.1. It runs on Python 3.6 to 3.9.

CHAPTER 3

Running a demo project

We assume this package is installed in your Python 3 environment.

Clone the project and go to `example` directory.

Now we need to create the database tables and an admin user. Run the following and follow the instructions:

```
$ ./manage.py migrate
$ ./manage.py createsuperuser
```

Now you may run the Django development server:

```
$ ./manage.py runserver
```

You should then be able to open your browser on http://127.0.0.1:8000 and see this app running.

CHAPTER 4

License

Django-dnsmanager uses the same license as Django (BSD-like) because we believe in open development. Please see LICENSE file for more details.

4.1 Models

dnsmanager.models.**A**
alias of *dnsmanager.models.AddressRecord*

dnsmanager.models.**AAAA**
alias of *dnsmanager.models.Ipv6AddressRecord*

class dnsmanager.models.**AddressRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Ipv4 Address record (abbreviated A) maps a hostname to a IPv4 address.

This format is defined in RFC 1035.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **address** (*GenericIPAddressField*) – Ipv4 address

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

address
Model field: IPv4 address

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

dnsmanager.models.**CAA**
alias of *dnsmanager.models.CertificationAuthorityAuthorizationRecord*

dnsmanager.models.**CNAME**
alias of *dnsmanager.models.CanonicalNameRecord*

class dnsmanager.models.**CanonicalNameRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Canonical name record (abbreviated CNAME) aliases one name to another.

This format is defined in RFC 1035. Please read <https://en.wikipedia.org/wiki/CNAME_record> for more details.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **c_name** (*CharField*) – Canonical name. This domain name will alias to this canonical name.

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

c_name
Model field: canonical name

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

class dnsmanager.models.**CertificationAuthorityAuthorizationRecord**(**args*, ***kwargs*)

Bases: *dnsmanager.models.Record*

A Certification Authority Authorization record (abbreviated CAA) constraints acceptable CAs for a host or domain.

This format is defined in RFC 6844.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **flags** (*PositiveIntegerField*) – Flags
- **tag** (*CharField*) – Tag
- **value** (*CharField*) – Value

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

TAGS = [('issue', 'issue'), ('issuewild', 'issue wildcard'), ('iodef', 'Incident objec

flags
Model field: flags

get_tag_display(**, field=<django.db.models.fields.CharField: tag>*)
Autogenerated: Shows the label of the *tag*

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

tag
Model field: tag

value
Model field: value

dnsmanager.models.**DNAME**
alias of *dnsmanager.models.DelegationNameRecord*

class dnsmanager.models.**DelegationNameRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Delegation name record (abbreviated DNAME), aliases a domain to the entire subtree of another domain.

This format is defined in RFC 6672. Please read <https://en.wikipedia.org/wiki/CNAME_record#DNAME_record> for more details.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **d_name** (*CharField*) – Delegation domain name. This domain name will alias to the entire subtree of that delegation domain.

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

d_name
Model field: delegation domain name

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

class dnsmanager.models.**Ipv6AddressRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Ipv6 Address record, or quad-A (abbreviated AAAA), maps a hostname to a IPv6 address.

This format is defined in RFC 3596. Please read <https://en.wikipedia.org/wiki/IPv6_address#Domain_Name_System> for more details.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **address** (*GenericIPAddressField*) – Ipv6 address

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

address
Model field: IPv6 address

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

dnsmanager.models.**MX**
alias of *dnsmanager.models.MailExchangeRecord*

class dnsmanager.models.**MailExchangeRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Mail Exchange record (abbreviated MX) maps a domain name to a list of message transfer agents for that domain.

This format is defined in RFC 1035 and 7505.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **preference** (*PositiveIntegerField*) – Preference
- **exchange** (*CharField*) – Exchange server

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

exchange
Model field: exchange server

preference
Model field: preference

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

dnsmanager.models.**NS**
alias of *dnsmanager.models.NameServerRecord*

class dnsmanager.models.**NameServerRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Name Server record (abbreviated NS) delegates a DNS zone to use the given authoritative name servers.

This format is defined in RFC 1035.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **nsdname** (*CharField*) – Name server

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

nsdname
Model field: name server

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

dnsmanager.models.**PTR**
alias of *dnsmanager.models.PointerRecord*

class dnsmanager.models.**PointerRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Pointer Resource record (abbreviated PTR) points a name to a canonical name.

Unlike a CNAME, DNS processing stops and just the name is returned. It is useful for implementing reverse DNS lookups.

This format is defined in RFC 1035.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.

- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **ptrdname** (*CharField*) – Pointer domain name

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

ptrdname
Model field: pointer domain name

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

class dnsmanager.models.**Record**(**args*, ***kwargs*)
Bases: polymorphic.models.PolymorphicModel

A generic DNS record of a zone.

This object should never be created directly as it is a polymorphic parent to all record types, but records can be retreived by requesting this object.

As Django DNSManager uses Django Polymorphic, a request of a record object will not return a record object but the polymorphic child which contains additionnal fields.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.

DNS_CLASSES = [('IN', 'IN (Internet)'), ('CS', 'CS (CSNET, obsolete)'), ('CH', 'CH (CH

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
Bases: django.core.exceptions.MultipleObjectsReturned

addressrecord
Model field: record ptr, accesses the *AddressRecord* model.

canonicalnamerecord
Model field: record ptr, accesses the *CanonicalNameRecord* model.

certificationauthorityauthorizationrecord
Model field: record ptr, accesses the *CertificationAuthorityAuthorizationRecord* model.

delegationnamerecord
Model field: record ptr, accesses the *DelegationNameRecord* model.

dns_class
Model field: class

get_dns_class_display(*, *field=<django.db.models.fields.CharField: dns_class>*)
Autogenerated: Shows the label of the *dns_class*

id
Model field: ID

ipv6addressrecord
Model field: record ptr, accesses the *Ipv6AddressRecord* model.

mailexchangerecord
Model field: record ptr, accesses the *MailExchangeRecord* model.

name
Model field: name

nameserverrecord
Model field: record ptr, accesses the *NameServerRecord* model.

pointerrecord
Model field: record ptr, accesses the *PointerRecord* model.

polymorphic_ctype
Model field: polymorphic ctype, accesses the `ContentType` model.

save(**args*, ***kwargs*)
Clean model on save

Without that the model does not get validated

servicerecord
Model field: record ptr, accesses the *ServiceRecord* model.

sshfingerprintrecord
Model field: record ptr, accesses the *SshFingerprintRecord* model.

startofauthorityrecord
Model field: record ptr, accesses the *StartOfAuthorityRecord* model.

textrecord
Model field: record ptr, accesses the *TextRecord* model.

ttl
Model field: Time To Live

zone
Model field: zone, accesses the *Zone* model.

zone_id
Model field: zone

dnsmanager.models.**SOA**
alias of *dnsmanager.models.StartOfAuthorityRecord*

dnsmanager.models.**SRV**
alias of *dnsmanager.models.ServiceRecord*

dnsmanager.models.**SSHFP**
alias of *dnsmanager.models.SshFingerprintRecord*

class dnsmanager.models.**ServiceRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Service record (abbreviated SRV) indicates the presence of a service.

It is a generalized service record instead of protocol-specific records such as MX.

This format is defined in RFC 2782. Please read <https://en.wikipedia.org/wiki/SRV_record> for more details.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **service** (*CharField*) – Service. The symbolic name of the desired service.
- **protocol** (*CharField*) – Protocol. The transport protocol of the desired service, usually either TCP or UDP.
- **priority** (*PositiveIntegerField*) – Priority. The priority of the target host, lower value means more preferred.
- **weight** (*PositiveIntegerField*) – Weight. A relative weight for records with the same priority, higher value means higher chance of getting picked.
- **port** (*PositiveIntegerField*) – Port
- **target** (*CharField*) – Target. The canonical hostname of the machine providing the service, ending in a dot.

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

port
Model field: port

priority
Model field: priority

protocol
Model field: protocol

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

service
Model field: service

target
Model field: target

weight
Model field: weight

class dnsmanager.models.**SshFingerprintRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A SSH Fingerprint record (abbreviated SSHFP) indicates the SSH public host key fingerprint of a host.

This format is defined in RFC 4255 and 6594.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **algorithm** (*PositiveIntegerField*) – Algorithm
- **type** (*PositiveIntegerField*) – Type
- **fingerprint** (*CharField*) – Fingerprint

ALGORITHMS = [(1, 'RSA'), (2, 'DSA'), (3, 'ECDSA'), (4, 'Ed25519')]

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

TYPES = [(1, 'SHA-1'), (2, 'SHA-256')]

algorithm
Model field: algorithm

fingerprint
Model field: fingerprint

get_algorithm_display(**, field=<django.db.models.fields.PositiveIntegerField: algorithm>*)
Autogenerated: Shows the label of the *algorithm*

get_type_display(**, field=<django.db.models.fields.PositiveIntegerField: type>*)
Autogenerated: Shows the label of the *type*

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

type
Model field: type

class dnsmanager.models.**StartOfAuthorityRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Start Of Authority record (abbreviated SOA) contains administrative information about the zone.

Every zone must have a SOA record to conform to the standard.

This format is defined in RFC 1035. Please read <https://en.wikipedia.org/wiki/SOA_record> for more details.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **mname** (*CharField*) – Master name server. Primary master name server for this zone.
- **rname** (*EmailField*) – Responsible email. Email address of the administrator responsible for this zone.
- **serial** (*BigIntegerField*) – Serial number. A slave name server will initiate a zone transfer if this serial is incremented.
- **refresh** (*BigIntegerField*) – Refresh. Number of seconds after which secondary name servers should query the master to detect zone changes.
- **retry** (*BigIntegerField*) – Retry. Number of seconds after which secondary name servers should retry to request the serial number from the master if the master does not respond.
- **expire** (*BigIntegerField*) – Expire. Number of seconds after which secondary name servers should stop answering request for this zone if the master does not respond.
- **minimum** (*BigIntegerField*) – Minimum. Time to live for purposes of negative caching.

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

clean()
Hook for doing any extra model-wide validation after clean() has been called on every field by self.clean_fields. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by NON_FIELD_ERRORS.

email_to_rname()
Convert email format to domain name format e.g. root@example.org to root.example.org

expire
Model field: expire

minimum
Model field: minimum

mname
Model field: master name server

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

refresh
Model field: refresh

retry
Model field: retry

rname
Model field: responsible email

serial
Model field: serial number

dnsmanager.models.**TXT**
alias of *dnsmanager.models.TextRecord*

class dnsmanager.models.**TextRecord**(**args*, ***kwargs*)
Bases: *dnsmanager.models.Record*

A Text record (abbreviated TXT) indicates arbitrary human-readable text.

This format is defined in RFC 1035 and 1464.

Parameters

- **id** (*AutoField*) – Id
- **polymorphic_ctype** (ForeignKey to ContentType) – Polymorphic ctype
- **zone** (ForeignKey to *Zone*) – Zone. This record will be applied on that zone.
- **name** (*CharField*) – Name. The domain name for which this record is valid, ending in a dot.
- **dns_class** (*CharField*) – Class. You shouldn't need anything else than IN.
- **ttl** (*PositiveIntegerField*) – Time to live. Limits the lifetime of this record.
- **record_ptr** (OneToOneField to *Record*) – Record ptr
- **data** (*TextField*) – Data

exception DoesNotExist
Bases: dnsmanager.models.DoesNotExist

exception MultipleObjectsReturned
Bases: dnsmanager.models.MultipleObjectsReturned

data
Model field: data

record_ptr
Model field: record ptr, accesses the *Record* model.

record_ptr_id
Model field: record ptr

class dnsmanager.models.**Zone**(**args*, ***kwargs*)
Bases: django.db.models.base.Model

A DNS Zone of a domain name contains all its records.

A DNS zone is represented by a zone text file that starts with the special DNS record type Start of Authority (SOA) and contains all records for the resources described within the zone.

This format is defined in RFC 1034 and RFC 1035. Please read <https://en.wikipedia.org/wiki/DNS_zone> for more details.

In Django DNSManager, zone text file can be generated by going to </dns/slug/>, "slug" being the value of the slug field in this object.

Parameters

- **id** (*AutoField*) – Id
- **name** (*CharField*) – Name
- **slug** (*SlugField*) – Slug. This zone will be accessible at /dns/{slug}/.

exception DoesNotExist
Bases: `django.core.exceptions.ObjectDoesNotExist`

exception MultipleObjectsReturned
Bases: `django.core.exceptions.MultipleObjectsReturned`

id
Model field: ID

name
Model field: name

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

record_set
Model field: zone, accesses the M2M *Record* model.

save(**args*, ***kwargs*)
Default value for slug

slug
Model field: slug

4.2 Views

class dnsmanager.views.**ZoneDetailView**(***kwargs*)
Bases: `django.contrib.auth.mixins.PermissionRequiredMixin`, `django.views.generic.detail.DetailView`

This view generates a zone file

model
alias of *dnsmanager.models.Zone*

permission_required = ('dnsmanager.view_zone', 'dnsmanager.view_record')

4.3 Integrations

4.3.1 Integration with Django Rest Framework

This app brings serializers and viewsets for Django Rest Framework. You can use those in your REST API like this,

```
from django.conf.urls import include, url
from rest_framework import routers
from dnsmanager.api import views

router = routers.DefaultRouter()
router.register(r'record', views.RecordViewSet)
router.register(r'zone', views.ZoneViewSet)

urlpatterns += [
    url(r'^api/', include(router.urls)),
]
```

4.3.1.1 Views

class dnsmanager.api.views.**RecordViewSet**(***kwargs*)
Bases: rest_framework.viewsets.ModelViewSet

queryset = PolymorphicQuerySet

serializer_class
alias of *dnsmanager.api.serializers.RecordPolymorphicSerializer*

class dnsmanager.api.views.**ZoneViewSet**(***kwargs*)
Bases: rest_framework.viewsets.ModelViewSet

queryset = QuerySet

serializer_class
alias of *dnsmanager.api.serializers.ZoneSerializer*

4.3.1.2 Serializers

class dnsmanager.api.serializers.**AAAASerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**ASerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**CAASerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**CNAMESerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**DNAMESerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**MXSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**NSSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**PTRSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**RecordPolymorphicSerializer**(**args*, ***kwargs*)

Bases: dnsmanager.api.polymorphic_serializer.PolymorphicSerializer

model_serializer_mapping = {<class 'dnsmanager.models.AddressRecord'>: <class 'dnsmana

class dnsmanager.api.serializers.**SOASerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**SRVSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**SSHFPSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**TXTSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class dnsmanager.api.serializers.**ZoneSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

F

G

I

M

N

O

P

Q

R

S

T

V

W

Z