Django Site Maintenance Documentation

Release 0.8

Stefano Apostolico

May 24, 2015

Contents

CHAPTER 1

Overview

Django Geo A Django application which manage administrative geographical data. It use Modified Preorder Tree Traversal, provided by django-mptt *https://github.com/django-mptt/django-mptt/*

CHAPTER 2

Table Of Contents

2.1 Installation

In order to install Django Geo simply use `pip`:

```
pip install django-geo
```

or `easy_install`:

```
easy_install django-geo
```

2.2 Configuration

After installation add geo to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    'geo',
)
```

2.3 Examples

2.3.1 Two levels

Country / City

```
italy = Country.objects.get(iso_code='IT')
city, __ = LocationType.objects.get_or_create(description='CITY')
roma_city, __  = Location.objects.get_or_create(country=italy,
                                                name ='Roma',
                                                type=city)
```

2.3.2 Three levels

Country / Regione / City

```
italy = Country.objects.get(iso_code='IT')
regione, __ = italy.administrativeareatype_set.get_or_create(name='Regione')


lazio, __ = AdministrativeArea.objects.get_or_create(country=italy,
                                                     name ='Lazio',
                                                     type=regione)

city, __ = LocationType.objects.get_or_create(description='CITY')
roma_city, __  = Location.objects.get_or_create(country=italy,
                                                name ='Roma',
                                                type=city,
                                                area=lazio)
```

2.3.3 Four levels

Represtent US administrative hierarchy:

Structure:

Country / State / County / Location

Es.

United States / New York / Columbia / Hudson

United States / New York / Albany / Albany

```
us = Country.objects.get(iso_code='US')
state, __ = italy.administrativeareatype_set.get_or_create(name='State')
county,__ = italy.administrativeareatype_set.get_or_create(name='County',
                                                           parent=state)
ny, __ = AdministrativeArea.objects.get_or_create(country=us,
                                                  name ='New York',
                                                  type=state)
columbia, __ = AdministrativeArea.objects.get_or_create(country=us,
                                                        name ='Columbia',
                                                        type=county,
                                                        parent=ny)
city, __ = LocationType.objects.get_or_create(description='CITY')
hudson, __  = Location.objects.get_or_create(country=us,
                                             name ='Hudson',
                                             type=city,
                                             area=columbia,
                                             is_administrative=True)
```

2.3.4 Five levels

Represtent Italy administrative hierarchy:

Structure:

Country / Regione / Provincia / Comune / Location

Es.

Italy / Lazio / Provincia di Roma / Comune di Roma / Roma

Italy / Lazio / Provincia di Roma / Comune di Roma / Ostia

Italy / Lazio / Provincia di Roma / Comune di Ciampino / Ostia

```
italy = Country.objects.get(iso_code='IT')

# create administrative structure for Italy
regione, __ = italy.administrativeareatype_set.create(name='Regione')
provincia,__ = italy.administrativeareatype_set.get_or_create(name='Provincia',
                                                              parent=regione)

comune, __= italy.administrativeareatype_set.get_or_create(name='Comune',
                                                           parent=provincia)


# add local administrative instances
lazio, __ = AdministrativeArea.objects.get_or_create(country=italy,
                                                     name ='Lazio',
                                                     type=regione)

roma_provincia, __ = AdministrativeArea.objects.get_or_create(country=italy,
                                                              name ='Provincia di Roma',
                                                              type=provincia,
                                                              parent=lazio)

roma_comune, __ = AdministrativeArea.objects.get_or_create(country=italy,
                                                           name ='Comune di Roma',
                                                           type=comune,
                                                           parent=roma_provincia)
city, __ = LocationType.objects.get_or_create(description='CITY')
roma_city, __  = Location.objects.get_or_create(country=italy,
                                                name ='Roma',
                                                type=city,
                                                area=roma_comune)
```

2.4 Models

2.4.1 Country

2.4.2 AdministrativeAreaType

2.4.3 AdministrativeArea

2.4.4 Location

2.4.5 Currency

2.5 Changelog

This sections lists the biggest changes done on each release.

2.6 0.8

- Django 1.7 support
- extra fields to Country model

2.7 0.7

- BACKWARD INCOMPATIBLE
- ALL MIGRATIONS RESET
- add UNRegion
- new UUIDField for each Model
- new LocationType Model

2.8 0.6

- fixed wrong/missed migrations
- extend test matrix
- fixes packaging

2.9 0.5

- switch to setuptools
- removed django-any, django-sample-data dependencies
- fixed migrations
- fixed compatibility django 1.5,16
- fixed tox
- added Makefile

2.10 0.4.2

- fixed natural key dependencies (ensures dumpdata puts json objects in an order for loaddata to load them)

2.11 0.4.1

Note: This version is backward incompatible

- fixed bugs causing dumpdata/loaddata not to work
- add uuid also to currency

2.12 0.4

Note: This version is backward incompatible

- uuid in models
- fix natural keys
- new model *LocationType*
- fixed natural_key issue in AdministrativeArea/AdministrativeAreaType
- reset migrations
- fixed dumpdata

2.13 Links

- Project home page: https://github.com/saxix/django-geo
- Issue tracker: https://github.com/saxix/django-geo/issues?sort
- Download: http://pypi.python.org/pypi/django-geo/
- Docs: http://readthedocs.org/docs/django-geo/en/latest/

2.13.1 Indices and tables

- genindex
- modindex
- search