geostore-backend-crud

Release 0.3.10

Nov 04, 2019

Contents

Terralego backend app

CHAPTER 1

Installation

1.1 Requirements

1.1.1 DATABASE

Minimum configuration :

- Postgresql 10
- PostGIS 2.4
- PgRouting 2.5

Recommended configuration :

- Postgresql 11
- PostGIS 2.5
- PgRouting 2.6

Your final django project should use django.contrib.gis.backend.postgis as default DATABASE backend

USING docker image :

Prebuilt docker image builded by makinacorpus

https://cloud.docker.com/u/makinacorpus/repository/docker/makinacorpus/pgrouting/general

1.1.2 SYSTEM REQUIREMENTS

For django

libpq-dev gettext

For geodjango

gdal-bin binutils libproj-dev

1.2 With pip

From Pypi:

```
pip install xxxxxxxxxx-xxxxxxxxxxxx
```

From Github:

```
pip install -e https://github.com/Terralego/django-geostore.git@master#egg=geostore
```

1.3 With git

```
git clone https://github.com/Terralego/django-geostore.git
cd django-geostore
python setup.py install
```

CHAPTER 2

Configuration

In your project :

settings

```
# install required apps
INSTALLED_APPS = [
    ...
    'django.contrib.gis',  # assume contrib.gis is installed
    ...
    'rest_framework',
    'rest_framework_gis',
    'geostore',
    ...
]

# optional overridable settings
INTERNAL_GEOMETRY_SRID = 4326 (can be changed for another SRID, should not be changed␣
↪after 1rst migration)

HOSTNAME = ''
TERRA_TILES_HOSTNAMES = [HOSTNAME, ]

MAX_TILE_ZOOM = 15
MIN_TILE_ZOOM = 10
```

urls

```
urlpatterns = [
    ...
    path('', include('geostore.urls', namespace='geostore')),
    ...
]
```

You can customize default url and namespace by including geostore.views directly

ADMIN :

you can disable and / or customize admin

- BACKWARD compatibility

settings to add :

```
import os

#####

HOSTNAME = os.environ.get('HOSTNAME', '')
TERRA_TILES_HOSTNAMES = [HOSTNAME, ]
```

CHAPTER 3

QUICK START

3.1 Manage layers

The simplest way to create a geographic data layer :

```
from geostore import GeometryTypes
from geostore.models import Layer

layer = Layer.objects.create(name='Mushroom spot',
                             geom_type=GeometryTypes.Point)
```

3.1.1 Geometry type validation

Layer support these geometry types :

Supported types

geostore.GeometryTypes

GeometryCollection = 7 LineString = 1 MultiLineString = 5 MultiPoint = 4 MultiPolygon = 6 Point = 0 Polygon = 3

Define a geometry type to layer to force feature geometry validation.

Without validation

```
from geostore.models import Layer, Feature
from geostore import GeometryTypes
from django.contrib.geos.geometries import GEOSGeometry

layer = Layer.objects.create(name='Mushroom spot 2')
```

(continues on next page)

(continued from previous page)

```
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)")

# ok
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("LINESTRING((0 0), (1 1))")
# ok too
```

With validation

```
from geostore.models import Layer, Feature
from geostore import GeometryTypes
from django.contrib.geos.geometries import GEOSGeometry

layer = Layer.objects.create(name='Mushroom spot 3',
                             geom_type=GeometryTypes.Point)
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)")

# ok
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("LINESTRING((0 0), (1 1))")
# validation error !
```

3.1.2 JSON schema definition / validation

You can use json schema definition to describe your data content, and improve feature properties validation.

https://json-schema.org/ https://rjsf-team.github.io/react-jsonschema-form/

```
from geostore.models import Layer, Feature
from geostore import GeometryTypes
from django.contrib.geos.geometries import GEOSGeometry

layer = Layer.objects.create(name='Mushroom spot 4',
                             geom_type=GeometryTypes.Point,
                             schema={
                               "required": ["name", "age"],
                               "properties": {
                                 "name": {
                                   "type": "string",
                                   "title": "Name"
                                 },
                                 "age": {
                                   "type": "integer",
                                   "title": "Age"
                                 }
                               }
                             })
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)")
# Validation Error ! name and age are required
```

(continues on next page)

(continued from previous page)

```
feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)",
                                 properties={
                                   "name": "Arthur",
                                 })
# Validation Error ! age is required

feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)",
                                 properties={
                                   "name": "Arthur",
                                   "age": "ten",
                                 })
# Validation Error ! age should be integer

feature = Feature.objects.create(layer=layer,
                                 geom=GEOSGeometry("POINT(0 0)",
                                 properties={
                                   "name": "Arthur",
                                   "age": 10
                                 })
# ok !
```

3.1.3 Vector tiles

geostore provide endpoint to generate and cache MVT based on your data.

You can access these tiles through Layer and LayerGroup features.

On layers

On group of layers

3.1.4 Data import

ShapeFile

GeoJSON

3.1.5 Data export

3.1.6 API endpoints