django-vectortiles

Release 1.0.0-beta2

Jean-Etienne Castagnede

May 24, 2023

CONTENTS

CHAPTER

ONE

INSTALLATION

1.1 Requirements

- You need to install geodjango required libraries (See here)

1.2 PostGIS 2.4+ backend usage

- You need a PostgreSQL database with PostGIS 2.4+ extension enabled. (See https://docs.djangoproject.com/en/stable/ref/contrib/gis/install/postgis/)
- You need to enable and use **django.contrib.gis.db.backends.postgis** database backend

```
pip install psycopg2
pip install django-vectortiles
```

1.3 Other database backend usages

```
pip install django-vectortiles[mapbox]
```

This will include sub-dependencies to generate vector tiles from mapbox_vector_tiles python library.

CHAPTER

TWO

USAGE

2.1 Simple model view

```
# in your app models.py
from django.contrib.gis.db import models

class Feature(models.Model):
    geom = models.GeometryField(srid=4326)
    name = models.CharField(max_length=250)

# in your view file
from django.views.generic import ListView
from vectortiles.postgis.views import MVTView
from yourapp.models import Feature


class FeatureTileView(MVTView, ListView):
    model = Feature
    vector_tile_layer_name = "features"  # name for data layer in vector tile
    vector_tile_fields = ('name',)  # model fields or queryset annotates to include in␣
↪tile
    # vector_tile_content_type = "application/x-protobuf"  # if you want to use custom␣
↪content_type
    # vector_tile_queryset = None  # define a queryset for your features
    # vector_tile_queryset_limit = None  # as queryset could not be sliced, set here a␣
↪limit for your features per tile
    # vector_tile_geom_name = "geom"  # geom field to consider in qs
    # vector_tile_extent = 4096  # tile extent
    # vector_tile_buffer = 256  # buffer around tile

# in your urls file
from django.urls import path
from yourapp import views


urlpatterns = [
    ...
    path('tiles/<int:z>/<int:x>/<int:y>', views.FeatureTileView.as_view(), name="feature-
↪tile"),
    ...
```

(continues on next page)

(continued from previous page)

```
]
```

2.2 Related model view

```
# in your app models.py
from django.contrib.gis.db import models


class Layer(models.Model):
    name = models.CharField(max_length=250)


class Feature(models.Model):
    geom = models.GeometryField(srid=4326)
    name = models.CharField(max_length=250)
    layer = models.ForeignKey(Layer, on_delete=models.CASCADE, related_name='features')

# in your views.py file
from django.views.generic import DetailView
from vectortiles.mixins import BaseVectorTileView
from vectortiles.postgis.views import MVTView
from yourapp.models import Layer


class LayerTileView(MVTView, DetailView):
    model = Layer
    vector_tile_fields = ('name', )

    def get_vector_tile_layer_name(self):
        return self.get_object().name

    def get_vector_tile_queryset(self):
        return self.get_object().features.all()

    def get(self, request, *args, **kwargs):
        self.object = self.get_object()
        return BaseVectorTileView.get(self,request=request, z=kwargs.get('z'), x=kwargs.
↪get('x'), y=kwargs.get('y'))


# in your urls file
from django.urls import path
from yourapp import views


urlpatterns = [
    ...
    path('layer/<int:pk>/tile/<int:z>/<int:x>/<int:y>', views.LayerTileView.as_view(),␣
↪name="layer-tile"),
    ...
```

(continues on next page)

(continued from previous page)

```
]
```

2.3 Django Rest Framework

```
# in your views.py file
from vectortiles.rest_framework.renderers import MVTRenderer


class FeatureAPIView(BaseVectorTile, APIView):
    vector_tile_queryset = Feature.objects.all()
    vector_tile_layer_name = "features"
    vector_tile_fields = ('name', )
    vector_tile_queryset_limit = 100
    renderer_classes = (MVTRenderer, )

    def get(self, request, *args, **kwargs):
        return Response(self.get_tile(kwargs.get('x'), kwargs.get('y'), kwargs.get('z')))

# in your urls file
urlpatterns = [
    ...
    path('features/tiles/<int:z>/<int:x>/<int:y>', FeatureAPIView.as_view(),
         name="feature-tile-drf"),
    ...
]

# or extending viewset

class FeatureViewSet(BaseVectorTile, viewsets.ModelViewSet):
    queryset = Feature.objects.all()
    vector_tile_layer_name = "features"
    vector_tile_fields = ('name', )
    vector_tile_queryset_limit = 100

    @action(detail=False, methods=['get'], renderer_classes=(MVTRenderer, ),
            url_path='tiles/(?P<z>\d+)/(?P<x>\d+)/(?P<y>\d+)', url_name='tile')
    def tile(self, request, *args, **kwargs):
        return Response(self.get_tile(x=int(kwargs.get('x')), y=int(kwargs.get('y')),␣
↪z=int(kwargs.get('z'))))

# in your urls file
router = SimpleRouter()
router.register(r'features', FeatureViewSet, basename='features')

urlpatterns += router.urls
```

then use http://your-domain/features/tiles/{z}/{x}/{y}.pbf

2.4 MapLibre Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
↪scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>MapBox / MapLibre example</title>
    <style>
        html, body {
            margin: 0;
            padding: 0;
        }
    </style>
    <link href='https://unpkg.com/maplibre-gl@2.4.0/dist/maplibre-gl.css' rel='stylesheet
↪'/>
    {#    <link href='https://watergis.github.io/maplibre-gl-legend/maplibre-gl-legend.
↪css' rel='stylesheet' />#}

</head>
<body>
<div id="map" style="width: 100%; height: 100vh"></div>
<script src='https://unpkg.com/maplibre-gl@2.4.0/dist/maplibre-gl.js'></script>
{#<script src="https://watergis.github.io/maplibre-gl-legend/maplibre-gl-legend.js"></
↪script>#}

<script>
    var map = new maplibregl.Map({
        container: 'map',
        hash: true,
        style: 'https://demotiles.maplibre.org/style.json', // stylesheet location
        center: [1.77, 44.498], // starting position [lng, lat]
        zoom: 8 // starting zoom
    });
    var nav = new maplibregl.NavigationControl({visualizePitch: true});
    map.addControl(nav, 'top-right');
    var scale = new maplibregl.ScaleControl({
        maxWidth: 80,
        unit: 'metric'
    });
    map.addControl(scale);
    map.on('load', function () {
        map.addSource('layers', {
            'type': 'vector',
            'url': '{% url "layer-tilejson" %}'
        });
        map.addLayer(
            {
                'id': 'background2',
```

(continues on next page)

(continued from previous page)

```
            'type': 'background',
            'paint': {
                'background-color': '#F8F4F0',
            }

        }
    );

    map.addLayer(
        {
            'id': 'foret_publique',
            'type': 'fill',
            'filter': ['==', ['geometry-type'], 'Polygon'],
            'source': 'layers',
            'source-layer': 'foret_publique',
            'paint': {
                'fill-color': '#64b646',
                'fill-opacity': 0.1
            }

        }
    );
    map.addLayer(
        {
            'id': 'parc_ou_reserve',
            'type': 'fill',
            'filter': ['==', ['geometry-type'], 'Polygon'],
            'source': 'layers',
            'source-layer': 'parc_ou_reserve',
            'paint': {
                'fill-color': '#64b646',
                'fill-opacity': 0.1
            }

        }
    );

    map.addLayer(
        {
            'id': 'troncon_hydrographique',
            'type': 'line',
            'source': 'layers',
            'source-layer': 'troncon_hydrographique',
            'paint': {
                'line-color': '#A7D4E8',
            }

        }
    );
    map.addLayer(
        {
            'id': 'surface_hydrographique',
```

(continues on next page)

(continued from previous page)

```
                'type': 'fill',
                'filter': ['==', ['geometry-type'], 'Polygon'],
                'source': 'layers',
                'source-layer': 'surface_hydrographique',
                'paint': {
                    'fill-color': '#A7D4E8',
                    'fill-opacity': 1
                }

            }
        );
        map.addLayer(
            {
                'id': 'troncon_de_voie_ferree',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_voie_ferree',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["!=", "nature", "Voie de service"],
                ],
                "paint": {
                    "line-color": "#bbb",
                    "line-width": {"base": 1.4, "stops": [[14, 0.4], [15, 0.75], [20,␣
↪2]]}
                }

            }
        );
        map.addLayer(
            {
                'id': 'troncon_de_voie_ferree_hatching',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_voie_ferree',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["!=", "nature", "Voie de service"],
                ],
                "paint": {
                    "line-color": "#bbb",
                    "line-dasharray": [0.2, 8],
                    "line-width": {"base": 1.4, "stops": [[14.5, 0], [15, 3], [20, 8]]}
                }

            }
        );
        map.addLayer(
            {
                'id': 'troncon_de_voie_ferree_service',
```

(continues on next page)

(continued from previous page)

```
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_voie_ferree',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["==", "nature", "Voie de service"],
                ],
                "paint": {
                    "line-color": "hsla(0, 0%, 73%, 0.77)",
                    "line-width": {"base": 1.4, "stops": [[14, 0.4], [20, 1]]}
                }

            }
        );
        map.addLayer(
            {
                'id': 'troncon_de_voie_ferree__service_hatching',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_voie_ferree',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["==", "nature", "Voie de service"],
                ],
                "paint": {
                    "line-color": "hsla(0, 0%, 73%, 0.68)",
                    "line-dasharray": [0.2, 8],
                    "line-width": {"base": 1.4, "stops": [[14.5, 0], [15, 2], [20, 6]]}
                }

            }
        );
        map.addLayer(
            {
                'id': 'troncon_de_route',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_route',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["!=", "nature", "Type autoroutier"],
                    ["!=", "nature", "Bretelle"],
                ],
                'layout': {
                    'line-cap': 'round',
                    'line-join': 'round'
                },
                'paint': {
                    'line-opacity': 1,
```

(continues on next page)

(continued from previous page)

```
                    'line-color': ['match', ['get', 'nature'], // get the property
                        'Type autoroutier', '#FF0000',              // if 'GP' then yellow
                        'XX', 'black',                // if 'XX' then black
                        'white'],
                    'line-width': {"base": 1.2, "stops": [[13.5, 0], [14, 2.5], [20, 11.
↪5]]}
                }
            }
        );
        map.addLayer(
            {
                'id': 'autoroute',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_route',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["==", "nature", "Type autoroutier"],
                ],
                'layout': {
                    'line-cap': 'round',
                    'line-join': 'round'
                },
                'paint': {
                    'line-opacity': 1,
                    'line-color': '#fc8',
                    'line-width': {"base": 1.2, "stops": [[6.5, 0], [7, 0.5], [20, 18]]}
                }
            }
        );
        map.addLayer(
            {
                'id': 'autoroute_bretelles',
                'type': 'line',
                'source': 'layers',
                'source-layer': 'troncon_de_route',
                "filter": [
                    "all",
                    ["==", "$type", "LineString"],
                    ["==", "nature", "Bretelle"],
                ],
                "layout": {"line-cap": "round", "line-join": "round"},
                "paint": {
                    "line-color": "#fc8",
                    "line-width": {
                        "base": 1.2,
                        "stops": [[12.5, 0], [13, 1.5], [14, 2.5], [20, 11.5]]
                    }
                }
            }
        );
```

(continues on next page)

(continued from previous page)

```
        map.addLayer(
            {
                'id': 'terrain_de_sport',
                'type': 'fill',
                'filter': ['==', ['geometry-type'], 'Polygon'],
                'source': 'layers',
                'source-layer': 'terrain_de_sport',
                'paint': {
                    'fill-color': '#E0EDD8',
                    'fill-opacity': 1
                }

            }
        );
        map.addLayer(
            {
                'id': 'batiment',
                'type': 'fill',
                maxzoom: 16,
                'filter': ['==', ['geometry-type'], 'Polygon'],
                'source': 'layers',
                'source-layer': 'batiment',
                'paint': {
                    'fill-color': '#F1EAD8',
                    'fill-opacity': 0.8
                }

            }
        );
        map.addLayer(
            {
                'id': 'batiment-contours',
                'type': 'line',
                minzoom: 14,
                maxzoom: 16,
                'filter': ['==', ['geometry-type'], 'Polygon'],
                'source': 'layers',
                'source-layer': 'batiment',
                'layout': {
                    'line-cap': 'round',
                    'line-join': 'round'
                },
                'paint': {
                    'line-opacity': 0.8,
                    'line-color': '#D6C3AC',
                    'line-width': 2,
                }

            }
        );
        map.addLayer(
            {
```

(continues on next page)

(continued from previous page)

```
                'id': 'batiment-3D',
                'filter': ['==', ['geometry-type'], 'Polygon'],
                minzoom: 16,
                'source': 'layers',
                'source-layer': 'batiment',
                'type': 'fill-extrusion',
                'paint': {
                    'fill-extrusion-color': '#F1EAD8',

                    // Use an 'interpolate' expression to
                    // add a smooth transition effect to
                    // the buildings as the user zooms in.
                    'fill-extrusion-height': [
                        'interpolate',
                        ['linear'],
                        ['zoom'],
                        15,
                        0,
                        15.05,
                        ['get', 'hauteur']
                    ],
                    'fill-extrusion-base': [
                        'interpolate',
                        ['linear'],
                        ['zoom'],
                        15,
                        0,
                        15.05,
                        2.5
                    ],
                    'fill-extrusion-opacity': 0.6
                }

            }
        );
                map.addLayer(
            {
                'id': 'commune',
                'type': 'line',
                'filter': ['==', ['geometry-type'], 'Polygon'],
                'source': 'layers',
                'source-layer': 'commune',
                'layout': {
                    'line-cap': 'round',
                    'line-join': 'round'
                },
                'paint': {
                    'line-opacity': 0.4,
                    'line-color': '#3636a8',
                    'line-width': 0.5,
                    'line-dasharray': [10, 10]
                }
```

(continues on next page)

(continued from previous page)

```
        }
    );
    map.addLayer(
        {
            'id': 'departement',
            'type': 'line',
            'filter': ['==', ['geometry-type'], 'Polygon'],
            'source': 'layers',
            'source-layer': 'departement',
            'layout': {
                'line-cap': 'round',
                'line-join': 'round'
            },
            'paint': {
                'line-opacity': 0.8,
                'line-color': '#479c47',
                'line-width': 0.8
            }

        }
    );
    map.addLayer(
        {
            'id': 'region',
            'type': 'line',
            'filter': ['==', ['geometry-type'], 'Polygon'],
            'source': 'layers',
            'source-layer': 'region',
            'layout': {
                'line-cap': 'round',
                'line-join': 'round'
            },
            'paint': {
                'line-opacity': 0.9,
                'line-color': '#b12929',
                'line-width': 1
            }

        }
    );
    map.addLayer(
        {
            "id": "commune_border",
            "type": "symbol",
            "source": "layers",
            "source-layer": "commune",
            "minzoom": 13,
            "layout": {
                "symbol-placement": "line",
                "symbol-spacing": 350,
                "text-field": "{nom}",
```

(continues on next page)

(continued from previous page)

```
                    "text-font": ["Noto Sans Italic"],
                    "text-letter-spacing": 0.2,
                    "text-max-width": 5,
                    "text-rotation-alignment": "map",
                    "text-size": 10
                },
                "paint": {
                    "text-color": "#3636a8",
                    "text-halo-color": "rgba(255,255,255,0.7)",
                    "text-halo-width": 1
                }
            }
        );
        map.addLayer(
            {
                "id": "commune_nom",
                "type": "symbol",
                "source": "layers",
                "source-layer": "commune_centre",
                "minzoom": 10,
                "maxzoom": 12,
                "layout": {
                    "symbol-placement": "point",
                    "symbol-spacing": 350,
                    "text-field": "{nom}",
                    "text-font": ["Noto Sans Italic"],
                    "text-letter-spacing": 0.2,
                    "text-max-width": 5,
                    "text-rotation-alignment": "map",
                    "text-size": 14
                },
                "paint": {
                    "text-color": "#3636a8",
                    "text-halo-color": "rgba(255,255,255,0.7)",
                    "text-halo-width": 1.5
                }
            }
        );
                map.addLayer(
            {
                "id": "eau_nom",
                "type": "symbol",
                "source": "layers",
                "source-layer": "troncon_hydrographique",
                "minzoom": 13,
                "layout": {
                    "symbol-placement": "line",
                    "symbol-spacing": 350,
                    "text-field": "{nom}",
                    "text-font": ["Noto Sans Italic"],
                    "text-letter-spacing": 0.2,
                    "text-max-width": 5,
```

(continues on next page)

(continued from previous page)

```
                    "text-rotation-alignment": "map",
                    "text-size": 14
                },
                "paint": {
                    "text-color": "#74aee9",
                    "text-halo-color": "rgba(255,255,255,0.7)",
                    "text-halo-width": 1.5
                }
            }
        );
        // Create a popup, but don't add it to the map yet.
        var popup = new maplibregl.Popup({
            closeButton: false,
            closeOnClick: false
        });

        map.on('mouseenter', 'commune_nom', function (e) {
            // Change the cursor style as a UI indicator.
            map.getCanvas().style.cursor = 'pointer';
            console.log(e.features);
            var coordinates = e.features[0].geometry.coordinates.slice();
            var description = `${e.features[0].properties.nom} (${e.features[0].
↪properties.population} hab.)`;

            // Ensure that if the map is zoomed out such that multiple
            // copies of the feature are visible, the popup appears
            // over the copy being pointed to.
            while (Math.abs(e.lngLat.lng - coordinates[0]) > 180) {
                coordinates[0] += e.lngLat.lng > coordinates[0] ? 360 : -360;
            }

            // Populate the popup and set its coordinates
            // based on the feature found.
            popup.setLngLat(coordinates).setHTML(description).addTo(map);
        });

        map.on('mouseleave', 'city-centroid', function () {
            map.getCanvas().style.cursor = '';
            popup.remove();
        });
    }
);
</script>
</body>
</html>
```

CHAPTER

THREE

CHANGELOG

3.1 1.0.0-beta

- Drop python 3.6 and Django 2.2
- Add python 3.11 and Django 4.2

** Breaking changes **

- Refactor PostGIS and Python (old named MapBox) backends usage. Use setting to set (default postgis)
- No DetailView anymore. As Tile can have many layers, declare VectorLayer on MTVView (one or many).
- Features
  - Native MVTRenderer for django-rest-framework
- Quality
  - Black-ified
  - iSort-ed
- Documentation
  - Add DRF and MapLibre examples

3.2 0.2.0 (2022-10-17)

- Possibly breaking change: * Base Mixin method get_tile use now class attributes for extent / buffer or clip_geom. Remove this parameters in your sub class method if needed.
- Bug fixes: * Correct usage for vector_tile_extent / vector_tile_buffer and vector_tile_clip_geom * Clipped geom is now working for mapbox
- Support Python 3.10 and django 4.1

3.3 0.1.0 (2021-02-25)

First beta release

- Add attribute to limit features in tile (unable to use a sliced queryset)

3.4 0.0.3 (2021-02-18)

- Delete useless Envelope transformation because django implicitly transform on intersects lookup (thanks to StefanBrand)
- Avoid useless queryset evaluation in some cases (thanks to StefanBrand)

3.5 0.0.2 (2021-02-12)

- Fix required 'fields' key in tilejson. Will be filled later
- Fix generated subquery to deal with DateField (thanks to StefanBrand)

3.6 0.0.1 (2020-10-22)

First Release

- **Generate Vector Tiles from django models**
  - in python
  - with PostGIS
- Generate associated TileJSON
- Default views to handle Vector tiles and tilejson

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search