

---

# **django-vectortiles**

***Release 0.2.0***

**Jean-Etienne Castagnede**

**Oct 17, 2022**



# CONTENTS

<b>1</b>	<b>INSTALLATION</b>	<b>1</b>
1.1	Requirements . . . . .	1
1.2	PostGIS 2.4+ backend usage . . . . .	1
1.3	Other database backend usages . . . . .	1
<b>2</b>	<b>USAGE</b>	<b>3</b>
2.1	Simple model view . . . . .	3
2.2	Related model view . . . . .	4
<b>3</b>	<b>Indices and tables</b>	<b>7</b>



## 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 [here](#))
- 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 subdependencies to generate vector tiles from mapbox\_vector\_tiles python library.



## 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)

]

## 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)

]
---



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`