

How to Django with MongoDB — The power of Django



Shameer Ahmed

Follow

Apr 30 · 3 min read



Due to the rise of the age of Big Data, NoSQL has become the industry norm for handling data for analyzing, real-time applications, and storage. NoSQL databases allow non-relational storage of data through which allows easy retrieval and storage of data in different ways.

SQL databases have the concept of SOLID and ACID whereas, NoSQL has no such concepts.

Considering the rise of NoSQL, Django should have NoSQL support such as MongoDB, but sadly there is no official support for MongoDB. There are few

open-source projects which provide MongoDB connector with Django. One such is the connector is djongo.

The djongo connector does not affect the original ORM of the Django ORM framework. The djongo connector not only translates SQL queries into MongoDB queries but also allows access to pymongo API which is official MongoDB support for Python.

Installation of Django

It can be accessed such as

```
pip install djongo
```

Add these in settings.py in Django Project

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
        # 'HOST': 'mongodb+srv://<username>:
<password>@mongodb_atlas',
        # 'USER': 'user',
        # 'PASSWORD': 'pass',
    }
}
```

The Host can be used to connect Django with Cloud Database of MongoDB with providing Username and Password for that Database

Run the following commands to generate schema of the MongoDB database

```
python manage.py makemigrations <app_name>
```

```
python manage.py migrate
```

Djongo mainly supports three types of data fields which are **Embedded field**, **Array Field**, **Array reference field**, and few others. To check them out click [here](#)

Also to access elements through the admin page of the Django app a form as to be declared with data fields such as

```
from djongo import models
from django import forms

class Blog(models.Model):
    name = models.CharField(max_length=100)
    tagline = models.TextField()

    class Meta:
        abstract = True

class BlogForm(forms.ModelForm):
    class Meta:
        model = Blog
        fields = (
            'name', 'tagline'
        )

class Entry(models.Model):
    blog = models.EmbeddedField(
        model_container=Blog,
        model_form_class=BlogForm
    )

    headline = models.CharField(max_length=255)
    objects = models.DjongoManager()
```

if you think you would like to serialize the data through the serializer then you can do that by using *rest_meets_djongo* serializer such as

```
from djongo import models
from rest_meets_djongo import serializers
from project.models import Entry
```

```
class EntrySerializer(serializers.DjangoModelSerializer):
    class Meta:
        model = Entry
        fields = ['headline', 'blog']
```

To use queries or sub-queries following syntax is required:

```
inner_qs=Blog.objects.filter(name__contains='django').values('name')
entries = Entry.objects.filter(blog__name__in=inner_qs)
```

Also, an important aspect is to serialized these models though serialization is not a necessity to step in working with django as it can directly access values from MongoDB using pymongo wrapper which can be accessed by adding *mongo_* to every command which exists in pymongo. Following is an example of aggeration using this method:

```
query=[
    {
        "$unwind": "$comments"
    },
    {
        "$group": {
            "_id": {
                "year": {
                    "$year": {
                        "$toDate": "$post_published"
                    }
                },
                "month": {
                    "$month": {
                        "$toDate": "$post_published"
                    }
                },
                "week": { "$week": { "$toDate": "$post_published" } }
            }
        },
        "post_sentiment": "$comments.sentiment",
```

```
        },
        "count": {
            "$sum": 1
        }
    },
    {
        "$sort": {
            "_id": -1
        }
    }
]
```

```
#this is used in view.py in Django
Posts.objects.mongo_aggregate(query)
```

Though Django has no official support for MongoDB as a framework for web application development, but *django* and other projects like this help to deal with solving complex problems and speed up the development cycle.

Disclaimer: *Django should not be used for production-level applications without a complete assessment of it for the project.*

References:

Django and MongoDB connector

Django is specifically meant to be used with the original Django ORM and MongoDB. Using the Django admin app...

nesdis.github.io

nesdis/django

Use MongoDB as a backend database for your Django project, without changing the Django ORM. Use the Djan...

github.com

Sign up for DDIntel

By Data Driven Investor


In each issue we share the best stories from the Data-Driven Investor's expert community. [Take a look](#)



Get this newsletter

Emails will be sent to hrishabh2203@gmail.com.

[Not you?](#)

Some rights reserved 

Mongodb

Django

Django Rest Framework

Data

Database

[About](#) [Help](#) [Legal](#)

Get the Medium app

