

Event

MongoDB is going on a world tour! Gather your team and head to your nearest MongoDB.local. View all cities >



How to Use Django with MongoDB

Not a MongoDB user yet?

[Register Free Today](#)

Python, the top programming language for data science, has always been a great match with MongoDB for building powerful applications. Django, the most popular Python web framework, is an ideal tool to build secure and easy-to-maintain applications using MongoDB.

Using MongoDB with Django is advantageous because:

- Every second, more and more **unstructured data** is generated from various sources like chats, real-time streams, feeds, and surveys.
- Data needs to be stored **securely** into a storage system that is scalable, performant, and not rigid, making it

easy to perform data **analysis** and visualizations.

- Data retrieval is easy because of the **simpler structure of storage**. That simplicity makes it easy to access even nested items and doesn't require complex joins.

MongoDB with Django satisfies the above criteria and provides much more. That is what we will learn in this tutorial.

Table of Contents

- **About MongoDB**
- **Connecting Django with MongoDB**
 - Django and MongoDB Using PyMongo
 - Django and MongoDB Using MongoEngine
 - Django and MongoDB Using Djongo
- **Django - MongoDB Tutorial**

[Get started free](#)[Learn more about Atlas](#)

About MongoDB

MongoDB is a flexible, schema-less, JSON-style, document-based database. Here's a side-by-side comparison showing the insertion of a medicine into a pharmacy database in SQL and MongoDB:

SQL

```
CREATE TABLE medicines (  
    medicine_id Varchar(11) NOT  
NULL,  
    common_name Varchar(15),  
    scientific_name Varchar(25),  
    available char(1),  
    category Varchar(10)  
)  
  
insert into  
medicines("RR000123456",  
"Paracetamol", "", "Y", "fever")
```

MongoDB

```
db.medicines.insertOne( {  
    medicine_id: "RR000123456",  
    common_name : "Paracetamol",  
    scientific_name : "",  
    available : "Y",  
    category: "fever"  
} )
```

Note that we inserted a record while creating the collection. In MongoDB, if the collection doesn't exist, it is created on the fly.

In SQL, if we forget to provide a **medicine_id**, the record will fail as it is the primary key (not null). Whereas

the default primary key in MongoDB is the `_id` field, we can insert the record without the `medicine_id`.

High availability, indexing, and non-reliance on complex joins are some other benefits of using MongoDB. Atlas provides security, scalability, and zero downtime patches and upgrades. Atlas is perfect to store big volumes of structured and unstructured data.

MongoDB Atlas

If you are going to work with big data, you will need to use a cloud-based service. This is where MongoDB Atlas's ability to support limitless clusters can help. You can easily deploy, operate, and scale MongoDB with Atlas. [Learn more about MongoDB Atlas.](#)

Get started with Atlas by following the [MongoDB Atlas tutorial instructions](#).

[Sign Up Free](#)

How Does Django Connect to MongoDB?

There are three ways to connect Django to MongoDB:

1. **PyMongo:** PyMongo is the standard driver through which MongoDB can interact with Django. It is the official and preferred way of using MongoDB with Python. PyMongo provides functionality to perform all the database actions like search, delete, update, and insert. Since PyMongo is available with PyPI, you can quickly install it using a pip command.
2. **MongoEngine:** MongoEngine is a Python Object-Document-Mapper. It's similar to Object-Relational-Mapper in relational databases. MongoEngine has a declarative API that is easy to learn and use.

3. **Djongo:** If you are using a relational database like SQL and want to migrate to MongoDB, for that you can use Djongo. Without changing the Django ORM, Djongo transpiles all the SQL queries to MongoDB syntax queries.

Which approach to connect to Django MongoDB is better? Let us explore this in the following sections.

Django and MongoDB Setup

To get the integration working, you should have a Django and MongoDB setup. If you have [Python](#) on your machine, you can install Django using pip. If you wish to install Django in a specific environment instead of the whole system, you can create a virtual environment. Use pip/pip3 depending on your Python version:

Install:

Windows:

```
pip install virtualenvwrapper-win
```

Mac OS / Linux:

```
pip install virtualenvwrapper
```

Create:

Windows:

```
mkvirtualenv MyProjectEnvt
```

Mac OS / Linux:

```
virtualenv MyProjectEnvt
```

Activate:

Mac OS / Linux

```
source MyProjectEnvt/bin/activate
```

Windows:

```
workon MyProjectEnvt
```

To deactivate the virtual environment, you can just type the command **deactivate**.

Now install Django using **pip install Django**.

To start a Django project, go to the folder where you want to start the project and use the below command:

```
django-admin startproject <project_name>.
```

For example,

```
C:\Users\myuser\project_files>django-admin startproject MyFirstDjangoProj  
C:\Users\myuser\project_files>cd MyFirstDjangoProj
```

To create an app, use the following command:

```
python manage.py startapp myfirstapp
```

If you are using the Python version ≥ 3.0 , replace python with python3 in your commands.

Inside the app, we can have many models that will be mapped to the collections and documents in MongoDB.

Once you start the project, all the files will be available in the project folder. Start the server using the **python manage.py runserver** command.

Your Django setup is now complete.

If you don't already have MongoDB set up, use [MongoDB Atlas](#) to make the most of cloud hosting. Atlas works seamlessly with all the major cloud providers.

[Return to Top](#)

Connect Django and MongoDB Using PyMongo

PyMongo is very efficient for writing JSON data to MongoDB and allows the use of MongoDB queries in the Python code itself. We can retrieve data in a dictionary like syntax using PyMongo.

Install PyMongo easily using the pip/pip3 command:

```
pip install pymongo[snapappy,gssapi,svr,tls]
```

If you are using a virtual environment (which you are!), you have to install pymongo in `..\venv\Lib\site-packages` folder.

Also, install dnspython for using mongodb+srv:// URLs with the command:

```
pip install dnspython
```

Using PyMongo, we can concurrently run multiple databases by specifying the right database name to the connection instance.

Let us create a sample pymongo session. There are two approaches for this:

1. We can create a client in the utils file that can be used by any view that wants to interact with MongoDB. Create a *utils.py* file in your project folder (same location as *manage.py*) and instantiate the client:

```
from pymongo import MongoClient
def get_db_handle(db_name, host, port, username, password):

    client = MongoClient(host=host,
                        port=int(port),
                        username=username,
                        password=password
                        )

    db_handle = client['db_name']
    return db_handle, client
```

This method can then be used in *./myfirstapp/view.py*.

2. Another approach to get the connection is to use the *connection_string*:

```
from pymongo import MongoClient
client = pymongo.MongoClient('connection_string')
db = client['db_name']
```

where


```
connection_string = mongodb+srv://<username>:<password>@<atlas cluster>
/<myFirstDatabase>?retryWrites=true&w=majority
```

For example,

```
makemyrx_db = client['sample_medicines']
#collection object
medicines_collection = makemyrx_db['medicinedetails']
```

You may have seen the Connection class used in other code samples or tutorials. Connection has been deprecated, so don't use it.

If you are on a default port and host, simply call *MongoClient()*. To connect to localhost, we can specify host and port explicitly as:

```
MongoClient('localhost', 27017)
```

or

use the URL format *MongoClient('mongodb://localhost: 27017/')*

Since we have created the client here, we need to comment the DATABASES section in the settings.py file. Comment the same using triple quotes.

[Return to Top](#)

Connect Django and MongoDB Using MongoEngine

MongoEngine is an [ORM layer](#) on top of PyMongo. So, you still need PyMongo (≥ 3.4) on your system to use MongoEngine.

Using MongoEngine to connect Django and MongoDB gives you fields like ListField and DictField to handle huge unstructured JSON data.

First, install MongoEngine using:

```
pip install mongoengine
```

As we have seen in the previous section, while using PyMongo, we have to comment the DATABASES section in settings.py. Then, to use MongoEngine, add the following:

```
import mongoengine
mongoengine.connect(db=db_name, host=hostname, username=username, password=pwd)
```

With MongoEngine, we have to define a schema in the models.py file of the Django application. MongoDB is schemaless. The schema is enforced only until application level, making any future changes fast and easy.

MongoEngine is similar to Django's default ORM, but with the following changes in model.py:

Django's ORM	MongoEngine
from django.db import models	from mongoengine import Document, fields or from mongoengine import *
Model	Document
models.CharField	fields.StringField()

The difference is that we were using *models*, which are replaced by *documents* and *fields* when MongoEngine is used.

There are many other tools written to work with [PyMongo](#).

[Return to Top](#)

Connect Django and MongoDB Using Django

Django is an improvement over PyMongo in that developers need not write lengthy queries. It maps Python objects to MongoDB documents, i.e., Object Document Mapping (ODM). Django ensures that only clean data enters the database. By performing integrity checks, applying validations, etc. with Django, there is no need to modify the existing Django ORM.

Install Django:

```
pip install django
```

Now, go to your project folder (example, **MyFirstDjangoProj**), and open settings.py file. You can edit it on Textpad, Python IDE, or any editor. Search for DATABASES, and change the settings to point to MongoDB. The ENGINE will be django and the database name (NAME) will be your MongoDB database name.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django',  
        'NAME': 'db-name',  
    }  
}
```

If your database is not on localhost or is secured, you should also fill in the CLIENT information like HOST, USERNAME, PASSWORD, etc.

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
        'ENFORCE_SCHEMA': False,
        'CLIENT': {
            'host': 'mongodb+srv://<username>:<password>@atlas cluster>/<myFirstDatabase>'
        }
    }
}
```

Make sure that the app-name is added in the INSTALLED_APPS setting of your settings.py:

```
INSTALLED_APPS = [
    'myfirstapp',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Now that we have the [Django project](#) (and app), you can create the collections in MongoDB using the commands:

```
python manage.py makemigrations <app-name>
```

```
python manage.py migrate
```

The collections (Django model in the app—note that we are talking about the *app* and not the *project*) will be created. You can check the same by opening Django Admin.

You can use the Admin GUI or insert data into collections manually.

To use the admin console, open your browser and go to <http://127.0.0.1:8000/admin> (or localhost). You should create a superuser to enter the admin console. If you do not have any models in your app, follow the [Django tutorial on how to create and register models](#).

If you want Django to go migration-free, set **ENFORCE_SCHEMA: False** in your database configuration. With this setting, the collections are created on the fly and Django won't transpile SQL statements into MongoDB commands.

[Return to Top](#)

Django and MongoDB Tutorial

(Feel free to code along or to download the full code from this [GitHub repo](#).)

In this quick tutorial, we will demonstrate how to use PyMongo to do simple CRUD operations. For this, let's create a PyMongo session:

```
import pymongo
#connect_string = 'mongodb+srv://<username>:<password>@<atlas cluster>/<myFirstDatabase>?retr

from django.conf import settings
```

```
my_client = pymongo.MongoClient(connect_string)

# First define the database name
dbname = my_client['sample_medicines']

# Now get/create collection name (remember that you will see the database in your mongodb cli
collection_name = dbname["medicinedetails"]

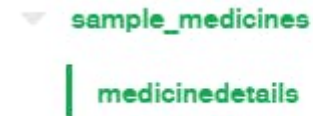
#let's create two documents
medicine_1 = {
    "medicine_id": "RR000123456",
    "common_name" : "Paracetamol",
    "scientific_name" : "",
    "available" : "Y",
    "category": "fever"
}
medicine_2 = {
    "medicine_id": "RR000342522",
    "common_name" : "Metformin",
    "scientific_name" : "",
    "available" : "Y",
    "category" : "type 2 diabetes"
}
# Insert the documents
collection_name.insert_many([medicine_1,medicine_2])
# Check the count
count = collection_name.count()
print(count)

# Read the documents
med_details = collection_name.find({})
# Print on the terminal
for r in med_details:
```

```
print(r["common_name"])
# Update one document
update_data = collection_name.update_one({'medicine_id': 'RR000123456'}, {'$set': {'common_name': 'Metformin'}})

# Delete one document
delete_data = collection_name.delete_one({'medicine_id': 'RR000123456'})
```

Next, you can connect to your MongoDB Atlas cluster and verify the same. Click on the collections and you can



spot the newly created database and the collection on the left:

On the right side, you will see one record (as we have deleted one):

A screenshot of the MongoDB Atlas 'Find' tab for the 'sample_medicines.medicinedetails' collection. The interface shows the collection name at the top, followed by statistics: 'COLLECTION SIZE: 147B', 'TOTAL DOCUMENTS: 1', and 'INDEXES TOTAL SIZE: 4KB'. Below this are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', and 'Aggregation'. A 'FILTER' input field contains the JSON query: {'filter': 'example'}. The 'QUERY RESULTS 1-1 OF 1' section displays a single document in a code-like format: {'_id': ObjectId('609a59ce2bfcdcfb1b8acc0d1'), 'medicine_id': 'RR000342522', 'common_name': 'Metformin', 'scientific_name': '', 'available': 'Y', 'category': 'type 2 diabetes'}.

sample_medicines.medicinedetails

COLLECTION SIZE: 147B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 4KB

Find Indexes Schema Anti-Patterns 0 Aggregation

FILTER {"filter": "example"}

QUERY RESULTS 1-1 OF 1

```
{
  "_id": ObjectId("609a59ce2bfcdcfb1b8acc0d1"),
  "medicine_id": "RR000342522",
  "common_name": "Metformin",
  "scientific_name": "",
  "available": "Y",
  "category": "type 2 diabetes"
}
```

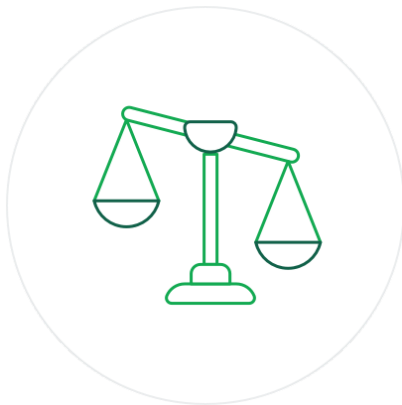
[Return to Top](#)

Next Steps

Now that we know the different ways to connect Django and MongoDB, we have to choose the right one for our project. The approaches have their own pros and cons.

For example, if you are starting from scratch, MongoEngine is a good way to go, as it can easily deal with a lot of unstructured data. If you have a lot of complex queries to write in your application, you should go with PyMongo.

Djongo is more suitable if you have a Django project that needs to migrate to MongoDB from another database, as that would require the least amount of code changes.

[Return to Top](#)

Ready to get started?

If you are going to work with big data, you will need to use a cloud-based service. This is where MongoDB Atlas's ability to support limitless clusters can help.

[Register Free Today](#)

FAQ

Can I use MongoDB with Django?



What is Django and MongoDB?




How does Django connect to MongoDB?



Which database is best for Django?



 English

About

Careers

Investor Relations

Legal Notices

Privacy Notices

Security Information

Trust Center

Support

Contact Us

Customer Portal

Atlas Status

Paid Support

Manage Cookies

Social



Github



Stack Overflow



LinkedIn



Youtube



Twitter



Twitch



Facebook

© 2023 MongoDB, Inc.