Working with Databases in Python 3

Using an ODM - MongoEngine

Douglas Starnes
Author / Speaker

@poweredbyaltnet | linktr.ee/douglasstarnes

Overview

Object Document Mapper

- Similar to an ORM, but maps to documents in a MongoDB database

MongoEngine

- Similar to the Django ORM API

Data modelling with document model classes

Embedded documents

Embedded lists

ORM Review

Object Relational Mapper

Create associations between Python objects and tables

Reduce or eliminate SQL code in your app

MongoEngine – an ODM

Object Document Mapper

Object Relational Mapper

MongoEngine – an ODM

Object Document Mapper

Object Relational Mapper

Relational databases are largely similar

They store data in two dimensional tables

Thus SQLAlchemy can use the same code with more than one database

NoSQL databases don’t follow a single pattern

MongoEngine works with MongoDB only

It will not work with other document databases or NoSQL database engines

Write mostly, or only, Python code

SQLAlchemy lets you avoid writing SQL code

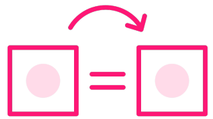
MongoEngine lets you avoid JSON

Work with Python objects

SQLAlchemy lets you avoid lists of tuples

MongoEngine lets you avoid lists and dictionaries

MongoEngine API



Similar to the Django ORM

MongoEngine is not a replacement for the Django ORM

MongoEngine used to support Django

MongoEngine does not add anything to the MongoDB database

MongoEngine Data Modeling

```
from mongoengine import Document

class Investment(Document):
```

MongoEngine Data Modeling

```
from mongoengine import Document, StringField, FloatField, DateTimeField, BooleanField

class Investment(Document):
```

MongoEngine Data Modeling

```
from mongoengine import *  # don't do this!

class Investment(Document):
```

MongoEngine Data Modeling

```
from mongoengine import Document
from mongoengine import fields

class Investment(Document):
```

MongoEngine Data Modeling

```
from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
```

MongoEngine Data Modeling

```
from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
    currency = fields.StringField(max_length=3)
```

MongoEngine Data Modeling

```
import datetime

from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
    currency = fields.StringField(max_length=3)
    amount = fields.FloatField(min_value=0.00001)
    timestamp = fields.DateTimeField(default=datetime.datetime.now)
```

MongoEngine Data Modeling

```
import datetime

from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
    currency = fields.StringField(max_length=3)
    amount = fields.FloatField(min_value=0.00001)
    timestamp = fields.DateTimeField(default=datetime.datetime.now)
    sell = fields.BooleanField(default=False)
```

MongoEngine Data Modeling

```
import datetime

from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
    # ...

bitcoin = Investment(coin="bitcoin", currency="USD", amount=1.0)
```

MongoEngine Data Modeling

```
import datetime

from mongoengine import Document
from mongoengine import fields

class Investment(Document):
    coin = fields.StringField(max_length=32)
    # ...

bitcoin = Investment(coin="bitcoin", currency="USD", amount=1.0)
bitcoin.save()
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)

gbp_investments = Investment.objects.filter(currency="GBP")
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)

gbp_investments = Investment.objects.filter(currency="GBP")

small_investments = Investment.objects.filter(amount__lt=1.0)
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)

gbp_investments = Investment.objects.filter(currency="GBP")

small_investments = Investment.objects.filter(amount__lt=1.0)

small_gbp_investments =
    Investment.objects.filter(currency="GBP").filter(amount__lt=1.0)
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)

gbp_investments = Investment.objects.filter(currency="GBP")

small_investments = Investment.objects.filter(amount__lt=1.0)

small_gbp_investments =
    Investment.objects.filter(currency="GBP").filter(amount__lt=1.0)

gbp_investment.update(amount=1.5)  # no need to call save()!
```

Querying MongoEngine Document Classes

```
for investment in Investment.objects:
    print(investment.coin)

gbp_investments = Investment.objects.filter(currency="GBP")

small_investments = Investment.objects.filter(amount__lt=1.0)

small_gbp_investments =
    Investment.objects.filter(currency="GBP").filter(amount__lt=1.0)

gbp_investment.update(amount=1.5)  # no need to call save()!

gbp_investment.delete()
```

Embedded Documents with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields
```

Embedded Documents with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
```

Embedded Documents with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
    currency = fields.StringField(max_length=3)
    description = fields.StringField()
    date_created =
        fields.DateField(default=datetime.datetime.now().date)
```

Embedded Documents with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
    currency = fields.StringField(max_length=3)
    description = fields.StringField()
    date_created =
        fields.DateField(default=datetime.datetime.now().date)

class Watchlist(Document):
    name = fields.StringField(max_length=256)
```

Embedded Documents with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
    currency = fields.StringField(max_length=3)
    description = fields.StringField()
    date_created =
        fields.DateField(default=datetime.datetime.now().date)

class Watchlist(Document):
    name = fields.StringField(max_length=256)
    metadata = fields.EmbeddedDocumentField(WatchlistMetadata)
```

Embedded Document Lists with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
    # ...

class WatchlistCoin(EmbeddedDocument):
    coin = fields.StringField(max_length=32)
    note = fields.StringField()
    date_added = fields.DateField(default=datetime.datetime.now().date)


class Watchlist(Document):
    name = fields.StringField(max_length=256)
    metadata = fields.EmbeddedDocumentField(WatchlistMetadata)
```

Embedded Document Lists with MongoEngine

```
from mongoengine import Document, EmbeddedDocument
from mongoengine import fields

class WatchlistMetadata(EmbeddedDocument):
    # ...

class WatchlistCoin(EmbeddedDocument):
    coin = fields.StringField(max_length=32)
    note = fields.StringField()
    date_added = fields.DateField(default=datetime.datetime.now().date)


class Watchlist(Document):
    name = fields.StringField(max_length=256)
    metadata = fields.EmbeddedDocumentField(WatchlistMetadata)
    coins = fields.EmbeddedDocumentListField(WatchlistCoin)
```

Creating Documents with MongoEngine

```
metadata = WatchlistMetadata(description="It's a watchlist", currency="USD")
```

Creating Documents with MongoEngine

```
metadata = WatchlistMetadata(description="It's a watchlist", currency="USD")

watchlist = Watchlist(name="My Watchlist", metadata=metadata, coins=[])
```

Creating Documents with MongoEngine

```
metadata = WatchlistMetadata(description="It's a watchlist", currency="USD")

watchlist = Watchlist(name="My Watchlist", metadata=metadata, coins=[])

watchlist.coins.append(WatchlistCoin(coin="dogecoin", note="It's a joke!"))
```

Creating Documents with MongoEngine

```
metadata = WatchlistMetadata(description="It's a watchlist", currency="USD")

watchlist = Watchlist(name="My Watchlist", metadata=metadata, coins=[])

watchlist.coins.append(WatchlistCoin(coin="dogecoin", note="It's a joke!"))

watchlist.save()
```

Querying Embedded Documents and Lists

```
import datetime

one_week = datetime.datetime.now().date() - datetime.timedelta(days=7)
recent_lists = Watchlist.objects(metadata__date_created__gte=one_week)
```

Querying Embedded Documents and Lists

```
import datetime

one_week = datetime.datetime.now().date() - datetime.timedelta(days=7)
recent_lists = Watchlist.objects(metadata__date_created__gte=one_week)


bitcoin_watchlist = Watchlist.objects(coins__coin="bitcoin").first()
```

Updating Embedded Document Lists

```
bears_watchlist = Watchlist.objects(name="Bears").first()

dogecoin = WatchlistCoin(coin="dogecoin", note="It's a joke!")
bears_watchlist.coins.append(dogecoin)
bears_watchlist.save()
```

Summary

MongoEngine

- Object Document Mapper

MongoEngine is syntactic sugar

You don’t have to work with raw lists and dictionaries (JSON)

The MongoEngine API is similar to the Django ORM

Every document is mapped to a Python document model class

Embedded documents and lists

Using the API and query operators to do CRUD with MongoEngine

Thank you!