MongoDB with MongoEngine

Using a document database like MongoDB is a common alternative to relational SQL databases. This pattern shows how to use MongoEngine, a document mapper library, to integrate with MongoDB.

A running MongoDB server and Flask-MongoEngine are required.

```
pip install flask-mongoengine
```

Configuration

Basic setup can be done by defining `MONGODB_SETTINGS` on `app.config` and creating a `MongoEngine` instance.

```
from flask import Flask
from flask_mongoengine import MongoEngine

app = Flask(__name__)
app.config['MONGODB_SETTINGS'] = {
    "db": "myapp",
}
db = MongoEngine(app)
```

Mapping Documents

To declare a model that represents a Mongo document, create a class that inherits from `Document` and declare each of the fields.

```
import mongoengine as me

class Movie(me.Document):
    title = me.StringField(required=True)
    year = me.IntField()
    rated = me.StringField()
    director = me.StringField()
    actors = me.ListField()
```

If the document has nested fields, use `EmbeddedDocument` to defined the fields of the embedded document and `EmbeddedDocumentField` to declare it on the parent document.

```
class Imdb(me.EmbeddedDocument):
    imdb_id = me.StringField()
    rating = me.DecimalField()
    votes = me.IntField()

class Movie(me.Document):
    ...
    imdb = me.EmbeddedDocumentField(Imdb)
```

Creating Data

Instantiate your document class with keyword arguments for the fields. You can also assign values to the field attributes after instantiation. Then call `doc.save()`.

```
bttf = Movie(title="Back To The Future", year=1985)
bttf.actors = [
    "Michael J. Fox",
    "Christopher Lloyd"
]
bttf.imdb = Imdb(imdb_id="tt0088763", rating=8.5)
bttf.save()
```

Queries

Use the class `objects` attribute to make queries. A keyword argument looks for an equal value on the field.

```
bttf = Movies.objects(title="Back To The Future").get_or_404()
```

Query operators may be used by concatenating them with the field name using a double-underscore. `objects`, and queries returned by calling it, are iterable.

```
some_theron_movie = Movie.objects(actors__in=["Charlize Theron"]).first

for recents in Movie.objects(year__gte=2017):
    print(recents.title)
```

Documentation

There are many more ways to define and query documents with MongoEngine. For more information, check out the official documentation.

Flask-MongoEngine adds helpful utilities on top of MongoEngine. Check out their documentation as well.