firestore-odm

Dec 24, 2019

Contents:

CHAPTER 1

Migrate from google.cloud.firestore

This document compares google-cloud-python-client firestore client library with firestore-odm.

1.1 Add data

It allows you to replace,

```
doc_ref = db.collection(u'users').document(u'alovelace')
doc_ref.set({
    u'first': u'Ada',
    u'last': u'Lovelace',
    u'born': 1815
})
```

with this,

```
user = User.new(doc_id="alovelace")
user.first = 'Ada'
user.last = 'Lovelace'
user.born = "1815"
user.save()
```

(Extra steps required to declare model. See quickstart for details.)

1.2 Read data

It allows you to replace,

```
users_ref = db.collection(u'users')
docs = users_ref.stream()
```

(continues on next page)

(continued from previous page)

```
for doc in docs:
    print(u'{} => {}'.format(doc.id, doc.to_dict()))
```

with this,

```
for user in User.all():
    print(user.to_dict())
```

1.3 Save data

It allows you to replace,

```
class City(object):
    def __init__(self, name, state, country, capital=False, population=0,
                 regions=[]):
        self.name = name
        self.state = state
        self.country = country
        self.capital = capital
        self.population = population
        self.regions = regions

    @staticmethod
    def from_dict(source):
        # ...

    def to_dict(self):
        # ...

    def __repr__(self):
        return(
            u'City(name={}, country={}, population={}, capital={}, regions={})'
            .format(self.name, self.country, self.population, self.capital,
                    self.regions))

cities_ref = db.collection(u'cities')
cities_ref.document(u'SF').set(
    City(u'San Francisco', u'CA', u'USA', False, 860000,
         [u'west_coast', u'norcal']).to_dict())
cities_ref.document(u'LA').set(
    City(u'Los Angeles', u'CA', u'USA', False, 3900000,
         [u'west_coast', u'socal']).to_dict())
cities_ref.document(u'DC').set(
    City(u'Washington D.C.', None, u'USA', True, 680000,
         [u'east_coast']).to_dict())
cities_ref.document(u'TOK').set(
    City(u'Tokyo', None, u'Japan', True, 9000000,
         [u'kanto', u'honshu']).to_dict())
cities_ref.document(u'BJ').set(
    City(u'Beijing', None, u'China', True, 21500000, [u'hebei']).to_dict())
```

with this,

```
def CityBase(DomainModel):
    _collection_name = "cities"

City = ClsFactory.create_customized(
        name="City",
        fieldnames=["name", "state", "country", "capital", "population", "regions"],
        auto_initialized=False,
        importable=False,
        exportable=True,
        additional_base=(CityBase,)
    )

City.new(
        doc_id='SF',
        name='San Francisco',
        state='CA',
        country='USA',
        capital=False,
        populations=860000,
        regions=['west_coast', 'norcal']).save()

# ...
```

(fieldname kwarg in ClsFactory to be implemented soon)

1.4 Get data

It allows you to replace,

```
doc_ref = db.collection(u'cities').document(u'SF')

try:
    doc = doc_ref.get()
    print(u'Document data: {}'.format(doc.to_dict()))
except google.cloud.exceptions.NotFound:
    print(u'No such document!')
```

with this,

```
sf = City.get(doc_id='SF')
if sf is not None:  # To be implemented soon
    print(u'Document data: {}'.format(doc.to_dict()))
else:
    print("No such document")
```

1.5 Simple queries

It allows you to replace,

```
docs = db.collection(u'cities').where(u'capital', u'==', True).stream()

for doc in docs:
    print(u'{} => {}'.format(doc.id, doc.to_dict()))
```

with this,

```
for city in City.where(capital=True):
    print(city.to_dict())
```

1.6 Query operators

It allows you to replace,

```
cities_ref = db.collection(u'cities')

cities_ref.where(u'state', u'==', u'CA')
cities_ref.where(u'population', u'<', 1000000)
cities_ref.where(u'name', u'>=', u'San Francisco'
```

with this,

```
City.where(state="CA")
City.where(population=('<', 1000000))
City.where(name=('>=', "San Francisco"))
```

1.7 Field name conversion

Sometimes, you want to have object attributes in “snake_case” and Firestore Document field name in “camelCase”. This is by default for flask-boiler. You may customize this conversion also.

Consider this example,

```
class CitySchema(Schema):
    city_name = fields.Raw()

    country = fields.Raw()
    capital = fields.Raw()


class MunicipalitySchema(CitySchema):
    pass


class StandardCitySchema(CitySchema):
    city_state = fields.Raw()
    regions = fields.Raw(many=True)


class City(DomainModel):
    _collection_name = "City"


Municipality = ClsFactory.create(
    name="Municipality",
    schema=MunicipalitySchema,
    base=City,
)
```

(continues on next page)

(continued from previous page)

```
StandardCity = ClsFactory.create(
    name="StandardCity",
    schema=StandardCitySchema,
    base=City
)

sf = StandardCity.create(doc_id="SF")
sf.city_name, sf.city_state, sf.country, sf.capital, sf.regions = \
    'San Francisco', 'CA', 'USA', False, ['west_coast', 'norcal']
sf.save()

la = StandardCity.create(doc_id="LA")
la.city_name, la.city_state, la.country, la.capital, la.regions = \
    'Los Angeles', 'CA', 'USA', False, ['west_coast', 'socal']
la.save()

dc = Municipality.create(doc_id="DC")
dc.city_name, dc.country, dc.capital = 'Washington D.C.', 'USA', True
dc.save()

tok = Municipality.create(doc_id="TOK")
tok.city_name, tok.country, tok.capital = 'Tokyo', 'Japan', True
tok.save()

beijing = Municipality.create(doc_id="BJ")
beijing.city_name, beijing.country, beijing.capital = \
    'Beijing', 'China', True
beijing.save()
```

object `la` saves to a document in firestore with "camelCase" field names,

```
{
    'cityName': 'Los Angeles',
    'cityState': 'CA',
    'country': 'USA',
    'capital': False,
    'regions': ['west_coast', 'socal'],
    'obj_type': "StandardCity",
    'doc_id': 'LA',
    'doc_ref': 'City/LA'
}
```

Similarly, you can query the objects with your local object attribute or firestore field name.

```
for obj in City.where(city_state="CA"):
    print(obj.city_name)
```

Or equivalently

```
for obj in City.where("cityState", "==", "CA"):
    print(obj.city_name)
```

CHAPTER 2

Context Management

In `__init__` of your project source root:

```
import os

from flask_boiler import context
from flask_boiler import config

Config = config.Config

testing_config = Config(app_name="your_app_name",
                        debug=True,
                        testing=True,
                        certificate_path=os.path.curdir + "/../your_project/config_
↪jsons/your_certificate.json")

CTX = context.Context
CTX.read(testing_config)
```

Note that initializing `Config` with `certificate_path` is unstable and may be changed later.

In your project code,

```
from flask_boiler import context

CTX = context.Context

# Retrieves firestore database instance
CTX.db

# Retrieves firebase app instance
CTX.firebase_app
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search