pyentity Documentation

Release

pyentity

February 20, 2015

Contents

1 Guide **3**
1.1 Tutorial 3

A simple object-document mapper written in python for motor(a driver for MongoDB) and to be used in tornado applications. Although, motor accomplish the hard task of provide async methods to access mongodb, we still have to get along with classic python dictionary which force us to create and maintain mapping of dicts-to-instance or instance-to-dicts.

Simple example using motor directly:

```
@gen.coroutine
def do_insert():
    # insert dict with the same attributes found in Product class
    product = {'name':'my cell phone','description':'blach phone','price':10.5}

    # call motor collection
    future = collection.insert(product)
    result = yield future
```

There's no problem if you prefer to work like this, but you always have to use some kind of mapping for any new added class in your model and ensure that no changes will break your existing code. In order to release you from boring task of accessing key/value to know something about your model, we suggest you to work in the following way:

Defining your class:

```
class Product(Entity):
    name = Str()
    description = Str()
    price = Float()

    def __init__(self, name="", description="", price=0.0):
        self.name = name
        self.description = description
        self.price = price
```

Writing some coroutine functions:

```
@gen.coroutine
def save_product():
    emanager = EntityManager(Product)
    product = Product('book', 'book of the year', 35.00)
    object_id = yield self.emanager.save(product)

@gen.coroutine
def find_product():
    emanager = EntityManager(Product)
    saved_product = yield self.emanager.find_one(object_id)
    name = saved_product.name
    description = saved_product.description
    price = saved_product.price
```

CHAPTER 1

Guide

1.1 Tutorial

We have been working on it