Co można wykrzesać z mongoengine?

Anna Warzecha

O mnie

- software engineer @ 10Clouds.com
- http://ania.cc
- anna.warzecha@gmail.com
- pink lover

JUST JOKING

mongoDB

- dynamiczne kolekcje dokumentów
- JSON/BSON (Binary JSON)
- zapytania w JavaScript
- ~~ACID (atomicity, consistency, isolation, durability)~~

mongoDB + python

- python
  - pymongo
    - minimongo
    - Homongolus
    - MongoKit
    - MongoAlchemy
    - Ming
    - mongoengine


django-nonrel

mongoengine - start

```
from mongoengine import *

connect('tumblelog')
```

Dokument

```
class User(Document):
    email = StringField(required=True)
    first_name = StringField(max_length=50)
    last_name = StringField(max_length=50)
```

```
{
  "_id" : ObjectId("50cf4c76c8c0c7a7f7fe9d7a"),
  "email" : "sampleuser@example.com",
  "first_name" : "Josh",
  "last_name" : "Bing"
}
```

Dziedziczenie

```
class Post(Document):
    meta = {'allow_inheritance': True}

    title = StringField(max_length=120,
        required=True)
    author = ReferenceField(User)

class TextPost(Post):
    content = StringField()

class ImagePost(Post):
    image_path = StringField()
```


LIKE A BOSS

Współdzielenie kolekcji

```
bookstore/
    __init__.py
    models.py ← class Books(Document)
    tests.py
    views.py
my_shelf/
    __init__.py
    models.py ← class Books(Document)
    tests.py
    views.py
```

EmbeddedDocument

```
class Comment(EmbeddedDocument):
    content = StringField()
    name = StringField(max_length=120)

class Post(Document):
    title = StringField(max_length=120,
        required=True)
    author = ReferenceField(User)
    tags = ListField(
        StringField(max_length=30)
    )
    comments = ListField(
        EmbeddedDocumentField(Comment)
    )
```

Tworzenie dokumentów

```
john = User(email='jdoe@example.com')
john.first_name = 'John'
john.last_name = 'Doe'
john.save()

post1 = TextPost(title='Fun with Mongo',
author=john)
post1.content = 'Took a look at MongoEngine
today, looks pretty cool.'
post1.tags = ['mongodb', 'mongoengine']
post1.save()
```

Odczyt dokumentów

```
# ten kod jest brzyyydki
for post in Post.objects:
    print post.title
    print '=' * len(post.title)

    if isinstance(post, TextPost):
        print post.content

    if isinstance(post, LinkPost):
        print 'Link:', post.link_url

    print
```

ONE DONE DOES NOT SIMPLY

USE MONGOENGINE

Filtrowanie

```
Post.objects.filter(title__icontains='Xmas')
Post.objects.fields(slice__comments=[5, 10])

Post.objects(__raw__={'tags': 'coding'})

# brak wyników
Post.objects.filter(__raw__={'_types':
'TextPost'})
# dokumenty typu TextPost
Post.objects.filter(class_check=False,
__raw__={'_types': 'TextPost'})
```

Magiczne QuerySety

```
class PostQuerySet(QuerySet):
    def texts(self):
        return self.filter(
            class_check=False,
            __raw__={'_types': 'TextPost'}
        )

    def texts_and_images(self):
        return self.filter(
            class_check=False,
            __raw__={'_types': {"$in":
                ['TextPost', 'ImagePost']
            }})
```

~~Full-Text Search~~

```
class Searcheable(object):
    _keywords = ListField(StringField())
    meta = {
        'allow_inheritance': True,
        'indexes': ['_keywords', ],
        'abstract': True
    }

    def pre_save(cls, sender, document, **kwargs):
        keywords = []
        for field in document._fields.keys():
            value = getattr(document, field, None)
            keywords.append(unicode(value))
        setattr(document, '_keywords', keywords)

pre_save.connect(Searcheable.pre_save)
```

class_check (?!)

```
def perform_search(qs, query):
    q_objs = None
    for q in query.split():
        kwargs = {'_keywords__icontains': q}
        if not q_objs:
            q_objs = Q(**kwargs)
        else:
            q_objs = q_objs and Q(**kwargs)
    qs = qs.filter(q_objs,
        class_check=qs._class_check)
    return qs
```


FAIL

Agregowanie

```
num_users = User.objects.count()
num_users = len(User.objects)

yearly_expense = Employee.objects.sum
('salary')

mean_age = User.objects.average('age')

freqs = Post.objects.item_frequencies('tags')
```

GridFS

```
img = ImageField(thumbnail_size=(48,48))

def get_img_display(self):
    thumbnail = self.img.thumbnail.read()
    img = thumbnail.encode('base64')
    return mark_safe(
        '<img src="%s"/>'
        % (self.img.format, img)
    )
```

MOAR

- map_reduce
- exec_js
- session storage i authentication backend dla Django

Dziękuję.