

26 ноября 2012 в 00:49

Использование MongoDB в Django

[Веб-разработка*](#), [MongoDB*](#), [Django*](#)

mongoDB

— документо-ориентированная система управления базами данных (СУБД) с открытым исходным

{name: "mongo", type: "DB"}

кодом, не требующая описания схемы таблиц. Написана на языке C++ и распространяется в рамках лицензии Creative Commons.

В последнее время становится довольно популярной и востребованной. И вот возникла идея использовать ее в связке с фреймворком Django. Собственно о чем далее и пойдет речь.

Для решения поставленной задачи мы будем использовать приложение `mongodb-engine`. Данное приложение тесно связано еще с несколькими приложениями, установкой которых мы и займемся вначале.

[Django-nonrel](#) — используется для поддержки NoSQL в Django.

```
pip install hg+https://bitbucket.org/wkornewald/django-nonrel
```

[djangotoolbox](#) — набор инструментов для работы с нереляционными базами данных, лишним не будет.

```
pip install hg+https://bitbucket.org/wkornewald/djangotoolbox
```

А теперь уже ставим и `mongodb-engine`:

```
pip install git+https://github.com/django-nonrel/mongodb-engine
```

Указываем нашу базу данных в `settings`:

```
DATABASES = {
    'default' : {
        'ENGINE' : 'django_mongodb_engine',
        'NAME' : 'my_database'
    }
}
```

При необходимости также можно указать `host`, `port`, `user`, `password`.

Данное приложение предоставляет два типа полей для хранения произвольных данных, не входящих в стандартную django модель.

`ListField`

Списки и им подобные, представление массивов в формате [BSON](#)

```
from djangotoolbox.fields import ListField

class Post(models.Model):
    ...
    tags = ListField()
```

```
>>> Post(tags=['django', 'mongodb'], ...).save()
>>> Post.objects.get(...).tags
['django', 'mongodb']
```

Вариант с указанием типа:

```
class Post(models.Model):
    ...
    edited_on = ListField(models.DateTimeField())
```

```
>>> post = Post(edited_on=['1010-10-10 10:10:10'])
>>> post.save()
>>> Post.objects.get(...).edited_on
[datetime.datetime([1010, 10, 10, 10, 10, 10])]
```

Данный тип поля удобно использовать для организации связи один-ко-многим:

```
from django_toolbox.fields import EmbeddedModelField, ListField

class Post(models.Model):
    ...
    comments = ListField(EmbeddedModelField('Comment'))

class Comment(models.Model):
    ...
    text = models.TextField()
```

`EmbeddedModelField` — используется для организации связей между моделями.

`DictField`

Второй тип поля `DictField`, который используется в BSON для объектов.

```
from django_toolbox.fields import DictField

class Image(models.Model):
    ...
    exif = DictField()
```

```
>>> Image(exif=get_exif_data(...), ...).save()
>>> Image.objects.get(...).exif
{'camera_model' : 'Spamcams 4242', 'exposure_time' : 0.3, ...}
```

Вариант с указанием типа:

```
class Poll(models.Model):
    ...
    votes = DictField(models.IntegerField())
```

```
>>> Poll(votes={'bob' : 3.14, 'alice' : '42'}, ...).save()
>>> Poll.objects.get(...).votes
{u'bob' : 3, u'alice' : 42}
```

Обновление данных

```
Post.objects.filter(...).update(title='Everything is the same')
```

Можно использовать для обновления оператор `$set`

```
.update(..., {'$set': {'title': 'Everything is the same'}})
```

А также функцию `F()`

```
Post.objects.filter(...).update(visits=F('visits')+1)
```

В результате получится что то такое:

```
.update(..., {'$inc': {'visits': 1}})
```

Использование низко-уровневых запросов

Если Вам не хватает возможностей Django ORM, можно использовать запросы к MongoDB минуя стандартный механизм.

`raw_query()` — принимает один аргумент, возвращает данные в виде стандартного Django `queryset`. Что хорошо для дальнейшей обработки данных.

Пример с гео данными, модель:

```
from djangotoolbox.fields import EmbeddedModelField
from django_mongodb_engine.contrib import MongoDBManager

class Point(models.Model):
    latitude = models.FloatField()
    longitude = models.FloatField()

class Place(models.Model):
    ...
    location = EmbeddedModelField(Point)

objects = MongoDBManager()
```

получим все точки рядом с конкретными координатами:

```
>>> here = {'latitude' : 42, 'longitude' : 3.14}
>>> Place.objects.raw_query({'location' : {'$near' : here}})
```

`raw_update()` — используется если нам недостаточно стандартных средств для обновления данных.

Модель:

```
from django_mongodb_engine.contrib import MongoDBManager

class FancyNumbers(models.Model):
    foo = models.IntegerField()

objects = MongoDBManager()
```

использование:

```
FancyNumbers.objects.raw_update({}, {'$bit' : {'foo' : {'or' : 42}}})
```

В данном примере выполняется побитовое `or` для каждого `foo` в базе.

На этом возможности данной связки не заканчиваются, но если пречислять все, то статья не оправдано затянется. Полное описание и примеры можно будет посмотреть по представленным ниже ссылкам.

Ссылки:

[MongoDB](#)

[mongodb-engine](#)

[Пример создания блога](#)

[GitHub](#)

[django](#), [mongodb](#), [nosql](#), [database](#) [изменить](#)

+14

10385

154

int22h 23,4

комментарии (14)

отслеживать новые: ☐ в почте ☒ в трекере



MechanisM, 26 ноября 2012 в 01:33 (комментарий был изменён) #

+12

На самом деле использование `django-nonrel` и `djangotoolbox` это не очень хороший способ. Ибо `django-nonrel` это не просто какой-то джанговский модуль для поддержки NoSQL а переделанный сам `django`.

Считаю что лучше уж использовать что-нить типа `mongoengine` + `django-mongonaut` (для админки) — тогда хоть останется родная Django.

А вообще ну не предусмотрена Django к использованию таких DB, так что если хочется такой экзотики, лучше уж Flask пользоваться и не мучить Django (сам уже себе шишек набил на этом).



[vak](#), 26 ноября 2012 в 12:23 #



0

а Flask развивается? тут что-то не очень активно: github.com/mitsuhiko/flask

а есть ли у вас ссылки на сравнение производительности Flask с другими фреймворками?



[MechanisM](#), 26 ноября 2012 в 12:45 #



+1

Развивается, и проекты написанные на нем растут как на дрожжах, как и всякие готовые модули спец под Flask.



[MechanisM](#), 26 ноября 2012 в 12:51 (комментарий был изменён) #



+2

Там особо активности то и не видно, ибо это микрофреймворк(не оч много кода) состоящий из других модулей(от этих-же разработчиков) Werkzeug, Jinja2 итд.

Да и если код не обновляется то это не значит что можно на нем поставить крест, может он настолько хорош что ничего не нужно менять(пока)? =))

Это джанго тащит в себе кучу функционала(до сих пор не понимаю почему бы обратно не выпилить оттуда gis в отдельный пакет, ибо этот функционал слишком специфичный), поэтому там много чего надо менять.



[vak](#), 26 ноября 2012 в 13:03 (комментарий был изменён) #



-1

можете поделиться собственным опытом в его масштабируемости и производительности?

3.bl. за наводку отблагодарил



[cadmi](#), 26 ноября 2012 в 19:41 #



0

Коротко говоря, django до flask в этом смысле как из Москвы до Пекина. Задним ходом. На четвереньках.



[dvs](#), 26 ноября 2012 в 15:58 #



0

Расскажите, пожалуйста, про шишки Django+mongo.



[qmax](#), 26 ноября 2012 в 11:01 #

+1

Django-nonrel — это же форк джанги.

К последним версиям джанги (1.3, 1.4) mongo-engine прикручивается само, правда без админки и model-forms.



[blackstone](#), 26 ноября 2012 в 12:54 #

+1

Я так понимаю этот пример сделан на базе туториала с официального сайта MongoDB [Write a Tumblelog Application with Django MongoDB Engine](#). Мне там не понравилось, что нужно тащить кучу библиотек и еще форк Django ставить.

Я давно думаю об использовании MongoDB + Django. Но пришел к мысли, что лучше использовать 2 базы — РСУБД и MongoDB. Для всего что укладывается в реляционную модель и входит в поставку Django использовать РСУБД, а MongoDB — только для тех сущностей, которые нужно уложить в документоориентированную модель.

И напрямую через pymongo.



[MechanisM](#), 26 ноября 2012 в 13:07 (комментарий был изменён) #



0

Тут есть [списочек приложений](#) использующих MongoDB(и через mongoengine и просто pymongo) в Django.

Не понимаю почему этим сайтом никто не пользуется? Ну понятное дело все любят сами писать, но ведь вся прелесть Django в том что уже существует куча готовых решений(надо только не лениться искать и применять).

Кстати, сам сайт djangorackages делал товарищ rudaппу, а потом на основе этого сравнения пакетов(гридов) он сделал свой стартапчик www.consumer.io/ — который, кстати, написан на Django + MongoDB. И приложение django-mongonaut(админка) было выпилено из кода consumer.io

Демка блора github.com/pydanny/django-mongonaut/tree/master/examples



[nogoody](#), 26 ноября 2012 в 13:00 #

0

Есть ли смысл тогда использовать Django? Если отказываемся от ORM проще Bottle взять, или CherryPy вместо Django



[int22h](#), 26 ноября 2012 в 13:25 #



0

От стандартной ORM не отказываемся.



[seriyPS](#), 26 ноября 2012 в 21:19 #

0

Django-nonrel уже портировали на что-то выше Django-1.3??

Вообще, если уж совсем приспичило работать из Django с MongoDB, возьмите Mongoengine. На порядок удобнее в работе.

Или вы ну прям совсем хотите от реляционных БД отказаться и использовать исключительно Mongo???



int22h, 26 ноября 2012 в 21:24 #



0

Ну прям совсем никто не отказывается. Реляционные базы вполне удовлетворяют в большинстве случаев.

«возьмите Mongoengine» — благодарю, обязательно посмотрю.

[Парадокс доказательства](#)

[Сканеры и копиры Xerox могут
менять цифры в документах
при копировании](#)

[Основная особенность наших
разработчиков](#)

[Все мозги в одном месте](#)

[Заказы для фрилансеров](#)

[Вакансии для айтишников](#)

[Уютная и дружелюбная](#)