Django Couchbase Documentation

Release 0.1

Aswin Kumar

Nov 20, 2017

Contents

Django Couchbase aims to provide ORM eqivalent to that of the django's default ORM for the couchbase database. With this package, accessing the couchbase database is not a headache anymore. With Django Couchbase, you can create the models the same way you do for relational databases.

Contents:

CHAPTER 1

Installation

1.1 Pre-requisite

It is assumes that you have a running couchbase instance. If you donot have it please download the latest version from http://www.couchbase.com/

1.2 Dependencies

couchbase==2.0.9 shortuuid==0.4.3 six==1.10.0 django-extensions==1.6.7 django-tastypie==0.13.3

1.3 Quick Install

Install django-couchbase package:

```
pip install django-couchbase
```

The following configuration settings are used for the package (you can use the set below for the fast installation):

```
CB_BUCKETS = {
    "MAIN_BUCKET" : '127.0.0.1/default'
}
```

Add `django_couchbase` to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    'django_couchbase',
)
```

CHAPTER 2

Getting Started with Django Couchbase

Django Couchbase is built on top of couchbase python library and was highly inspired from django_cbtools and Django non-rel . Since *django_cbtools* is more sync_gateway focused package, this package would not require sync_gateway to get started.

2.1 Writing a Model

2.1.1 Models

There are two types of base classes that support different purposes.

- `CBModel`
- `CBNestedModel`

The CBModel is the class that forms the root of the JSON document. CBNestedModel can ony be nested. You cannot save it or retrive it directly.

2.1.2 Fields

Below are the fields that we are going to use for NoSql- specific functionalities.

- `ListField`

  This field is used to create the array inside the JSON document.

- `EmbeddedModelField`

  This field refers to another class that when serialized creates the nested JSON under the specified property.

- `ModelReferenceField`

  This field is like the usual foreign key field that stores the corresponding document elsewhere and only holds the id in that JSON document.

Let us have a look at the example before we actually dive into more code. Note the above said class names and fields:

```
from django_couchbase.models import CBModel,CBNestedModel
from django_couchbase.fields import PartialReferenceField, ModelReferenceField
from djangotoolbox.fields import ListField, EmbeddedModelField, DictField

class Article(CBNestedModel):
    class Meta:
        abstract = True

    doc_type = 'article'
    id_prefix = 'art'

    title = models.CharField(max_length=45, null=True, blank=True)

class Blog(CBNestedModel):
    class Meta:
        abstract = True

    doc_type = 'blog'
    id_prefix = 'blg'

    url = models.CharField(max_length=45, null=True, blank=True)
    articles = ListField(EmbeddedModelField(Article))

class Publisher(CBModel):
    class Meta:
        abstract = True

    doc_type = 'publisher'
    id_prefix = 'pub'
    bucket = "MAIN_BUCKET"

    name = models.CharField(max_length=45, null=True, blank=True)

class Book(CBModel):
    class Meta:
        abstract = True

    doc_type = 'book'
    id_prefix = 'bk'
    bucket = "MAIN_BUCKET"

    name = models.CharField(max_length=45, null=True, blank=True)
    pages = models.IntegerField()
    publisher = ModelReferenceField(Publisher)

class Address(CBModel):
    class Meta:
        abstract = True

    doc_type = 'address'
    id_prefix = 'addr'
    bucket = "MAIN_BUCKET"

    street = models.CharField(max_length=45, null=True, blank=True)
    city = models.CharField(max_length=45, null=True, blank=True)

class Author(CBModel):
```

```
    class Meta:
        abstract = True

    doc_type = 'author'
    id_prefix = 'atr'
    bucket = "MAIN_BUCKET"

    name = models.CharField(max_length=45, null=True, blank=True)
    blog = EmbeddedModelField(Blog)
    books = ListField(ModelReferenceField(Book))
    address = ModelReferenceField(Address)
```

Enough. Let me explain the code above.

- As stated above note the classed were inherited from the `CBModel` and `CBNestedModel`. You can also use relational databases in other models by extending from `models.Model`.
- `abstract = True` should be added to all classes that has the parent of `CBModel` or `CBNestedmodel` to avoid making migrations to those classes and ading them in relational database schema.
- `doc_type = 'article'` is the field you have to define. This is the way Django Couchbase stores the type of the objects. This value is stored in the database.
- `id_prefix = 'atl'` this is an optional prefix for the `uid` of the document. Having prefix for the `uid` help a lot to debug the application. For example you can easily define type of the document having just its `uid`. Very useful.

2.2 Creating Documents

You can create the document in the following way:

```
# Creating two articles.
article = Article(title = "New Article")
article2 = Article(title = "Second Article")

# Create a blog that has both the article nested in it
blog = Blog(url = "4sw.in", articles = [article, article2])

# Create two publishers
pub = Publisher(name = "Famous Publications")
pub2 = Publisher(name = "Much more Famous Publications")

# Add the publishers as the reference
book = Book(name = "First Book", pages = 250, publisher = pub)
book2 = Book(name = "Second Book", pages = 340, publisher = pub2)

# Create the address  document
address = Address(street = "Anna Nagar", city = "Chennai")

# embed blog, books, address in author document
author = Author(name = "Aswin", blog = blog, books = [book, book2], address=address)

# save all the above models in the database
author.save()
```

You can use them in any combiations you want. Like

```
* ListField
* EmbeddedField
* ModelReferenceField
* ListField(EmbeddedModelField)
* ListField(ModelReferenceField)
```

2.3 Retriving Documents

Document retrival is more similar process:

```
author = Author('atl_0a1cf319ae4e8b3d5f8249fef9d1bb2c')
print author
```

2.4 Loading related documents

This is to retrive the documents in the `ModelReferenceField`.

CHAPTER 3

ToDo

CHAPTER 4

Indices and tables

- genindex
- search