Frink Documentation

Release 0.0.3

Hactar

April 13, 2016

Contents

CHAPTER 1

About

Frink is a super basic ORM-like thing for using RethinkDB in Flask, built on top of Schematics. It has built in compatibility with *Flask-Security*.

Warning: Frink is currently pre-alpha and extremely likely to change. However, it is stable enough in the Flask-Security datastore that you could probably use it to enable Flask-Security to work with RethinkDB.

1.1 Install

```
pip install frink
```

If you are contributing to the development of Frink, install it as an editable package from your own clone of the git repo. That way you can commit and push you edits.

```
git clone git@github.com:hactar-is/frink.git lib/frink
pip install -e lib/frink
```

1.2 Quickstart

1.2.1 Flask

Frink is designed to be used with the Application Factory pattern in Flask.

```
from frink.connection import RethinkDB

db = RethinkDB()
```

Then in your application factory, call init_app on the DB object.

```
def create_app():
    ...
    db.init_app(app)
```

1.2.2 Flask-Security

Frink includes `FrinkDatastore` and `FrinkUserDatastore` for Flask-Security compatibility.

Define your `User` and `Role` models.

```
import datetime
from schematics.types.base import (
    StringType, BooleanType, DateTimeType, IntType
)

from schematics.types.compound import (
    ListType, ModelType
)

from flask.ext.security import UserMixin, RoleMixin

from frink.base import BaseModel
from frink.orm import ORMMeta


class Role(BaseModel, RoleMixin):

    __metaclass__ = ORMMeta

    name = StringType()
    description = StringType()


class User(BaseModel, UserMixin):

    __metaclass__ = ORMMeta

    email = StringType()
    password = StringType()
    active = BooleanType(default=True)
    confirmed_at = DateTimeType()
    last_login_at = DateTimeType(default=datetime.datetime.now)
    current_login_at = DateTimeType(default=datetime.datetime.now)
    registered_at = DateTimeType()
    last_login_ip = StringType()
    current_login_ip = StringType()
    login_count = IntType()

    roles = ListType(ModelType(Role))
```

Then in your application factory, initialise this...

```
from frink.datastore import FrinkUserDatastore
from .users.models import User, Role

def create_app():
    ...
    user_datastore = FrinkUserDatastore(db, User, Role)
    security.init_app(app, user_datastore)
    app.user_datastore = user_datastore
```

1.3 ORM

The ORM part of Frink gives you very basic querying abilities. For anything a little more complex you should probably just drop down to *ReQL* anyway.

1.3.1 Querying

Get a single instance by id.

```
User.query.get('9353b884-591b-404f-a4e2-30334d5ad335')
```

Get all instances.

```
User.query.all()
```

Get all instances, ordered and limited.

```
User.query.all(order_by='firstname', limit=10)
```

Filtering

Get a single instance, that is the first to match the filter.

```
User.query.first(firstname='Jeff')
```

Get a list of results matching kwargs filters.

```
User.query.filter(active=True, firstname='Jeff')
```

Order the results by firstname ascending.

```
User.query.filter(active=True, firstname='Jeff', order_by='<name')
```

Order the results by firstname descending.

```
User.query.filter(active=True, firstname='Jeff', order_by='>name')
```

Limit the results.

```
User.query.filter(active=True, firstname='Jeff', order_by='>name', limit=10)
```

Find all instances that match one column / value.

```
User.query.find_by(column='firstname', value='Jeff')
```

Ordered and limited.

```
User.query.find_by(column='firstname', value='Jeff', order_by='lastname', limit=1)
```

Get a single instance matched on column / value.

```
User.query.get_by(column='firstname', value='Jeff')
```

Todo

Add an `offset` parameter to all methods that have a `limit` argument for use with pagination.

1.3.2 ReQL

The models all contain references to the database and the table that they're stored in, just in case you ever need to dynamically create a ReQL query. It's also fairly useful even if you're writing them by hand.

```
r.db(User._db).table(User._table).filter({"firstname": "Jeff"}).run(conn)
```

1.4 API

1.4.1 Datastores

1.4.2 ORM

1.5 Change Log

1.5.1 0.0.3

- Improved test coverage
- Fixed bug in ordering on filter

1.5.2 0.0.2

- Added some tests
- Fixed order of filter and limit
- Updated to require rethinkdb==2.3.0 due to a bug in 2.2.6

CHAPTER 2

Contributing

If you think Frink could be useful for you, you could help get it to some kind of stability by getting involved. Contributors welcome.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search