Django Changeflow Documentation

Release 0.1

synw

September 01, 2016

CHAPTER 1

Install

```
cd my_project
git clone https://github.com/synw/django-changefeed.git && mv django-changefeed/changefeed . && rm -r
pip install rethinkdb celery
```

Then add `'changefeed',` to INSTALLED_APPS.

Now create your database and table in the Rethinkdb client.

1.1 Settings

1.1.1 Required settings

```
SITE_SLUG = "mysite"
```

This is required for internal prefixing

1.1.2 Optional settings

You can set a default database and default table to be used:

```
# default database is set to SITE_SLUG+'_changefeed'
CHANGEFEED_DB = "mydb"
# default table is set to "changefeed"
CHANGEFEED_TABLE = "mytable"
```

The worker is verbose by default. To silence it use this:

```
CHANGEFEED_VERBOSE = False
```

The feed listener is launched by default. If you wish not to listen to the changefeed:

```
CHANGEFEED_LISTEN = False
```

1.2 Run

Run Rethinkdb and Launch a Celery worker:

```
celery -A project_name worker  -l info --broker='redis://localhost:6379/0'
# or any option you want
```

Push a document to Rethinkdb

```
from changefeed.tasks import push_to_flow, push_to_db

data = {"field_name":"field_value"}
# this is to push to the default db and table
push_to_feed.delay(data)

# this is to push to a db and table of your choice
push_to_db.delay("database_name", "table_name", data)
```

CHAPTER 3

Handle the changes events

To handle the data changes create a r_handlers.py file in your project directory (where settings.py is).

3.1 Set the ReQL query

The listener will use by default the following ReQL query, listening to all changes in the table:

```
r.db(database).table(table).changes()
```

You can set a custom ReQL query for the listener in r_handlers.py with a r_query function:

```
def r_query():
     return r.db("mydb").table("mytable").pluck('message').changes()
```

3.2 Write the handlers

```
# in settings.py
CHANGEFEED_LISTEN = True
```

In your root directory (SITE_SLUG) create a r_handlers.py file and a feed_handlers function to do whatever you want, like to broadcast the data to a websocket:

```
# this function will be triggered on every change in the Rethinkdb data
def feed_handlers(database, table, change):
    print database
    print table
    print str(change['old_val'])
    print str(change['new_val'])
    return
```

You can also manage handlers directly from your apps by creating a file with a feed_handlers function, and launch a listener from the __init__.py file of your app:

```
from changefeed.tasks import feed_listener
feed_listener.delay("database", "table", "mymodule.r_handlers")
```

CHAPTER 4

Example

Example hello world:

Make a “testdb” and a “testtable” in the Rethinkdb client.

```
# in settings.py
CHANGEFEED_DB = "testdb"
CHANGEFEED_TABLE = "testtable"

CHANGEFEED_HANDLERS = ['mymodule.r_handlers']
# or just use a r_handlers.py file in your main app directory

# in mymodule/r_handlers.py
def feed_handlers(database, table, change):
     message = change['new_val']['message']
     print message
  return

 # optionaly define a ReQL query in mymodule/r_handlers.py
 def r_query():
     r.db("testdb").table("testtable").pluck('message').changes()

# anywhere in your code
from changefeed.tasks import push_to_feed

push_to_feed({"message":"Hello world"})
```