django-balancer Documentation

Release 0.3

Brandon Konkle

August 17, 2015

Contents

Contents:

CHAPTER 1

Installation

Use pip to install the module:

```
$ pip install django-balancer
```

Then, add the desired router to your DATABASE_ROUTERS setting:

```
DATABASE_ROUTERS = ['balancer.routers.WeightedRandomRouter']
```

Finally, add any configuration settings needed for the router chosen:

```
DATABASE_POOL = {
    'default': 2,
    'db02': 1,
    'db03': 1,
}
```

CHAPTER 2

Routers

2.1 RandomRouter

Randomly selects from a pool of database for both reads and writes. Useful for replication configurations where all nodes act as masters.

2.1.1 Required Settings

- *DATABASE_POOL*

2.2 WeightedRandomRouter

This router applies weighting to the random selection. This would be useful for configurations where all nodes act as masters, but you'd like some nodes to get more traffic than others.

2.2.1 Required Settings

- *DATABASE_POOL*

2.3 RoundRobinRouter

A router that cycles over a pool of databases in order, evenly distributing the load. The pool is shuffled upon initialization of the class, to avoid overloading the master database at startup.

2.3.1 Required Settings

- *DATABASE_POOL*

2.4 WeightedMasterSlaveRouter

This router allows you to use the database pool for reads, but only the database you designate as master for writes. This is useful for master/slave configurations. If you don't include the master database in the pool, it will only be used for writes. Uses weighted random selection.

2.4.1 Required Settings

- *DATABASE_POOL*
- *MASTER_DATABASE*

2.5 RoundRobinMasterSlaveRouter

Same as above, but using round robin database selection instead.

2.6 PinningWMSRouter

This is a master/slave router that uses weighted random selection and pins reads to the master for a user after that user has executed a write to the database. This is useful for replication configurations where there is a noticeable amount of lag between a write to master and the propagation of that data to the slave databases.

To use this router, you also need to use one of the included pinning middleware classes. PinningSessionMiddleware uses the Django sessions contrib app, and PinningCookieMiddleware uses a cookie.

2.6.1 Required Settings

- *DATABASE_POOL*
- *MASTER_DATABASE*

2.6.2 Optional Settings

- *MASTER_PINNING_KEY*
- *MASTER_PINNING_SECONDS*

2.7 PinningRRMSRouter

Same as above, but using round robin database selection instead.

CHAPTER 3

Settings

3.1 DATABASE_POOL

Can be either a list of database names to include in the pool, or a dict mapping the databases to their weights.

Example:

```
DATABASE_POOL = {
    'default': 2,
    'db02': 1,
    'db03': 1,
}
```

3.2 MASTER_DATABASE

The database that should be used for all writes. Expects a string.

3.3 MASTER_PINNING_KEY

The name of the session variable or cookie used by the pinning middleware. Expects a string.

Defaults to: `'master_db_pinned'`

3.4 MASTER_PINNING_SECONDS

The number of seconds to direct reads to the master database after a write. Expects an integer.

Defaults to: `5`

CHAPTER 4

Indices and tables

- genindex
- modindex
- search