# django-anon Documentation

Tesorio

Dec 18, 2020

# Contents

**1 Table of Contents** **3**
1.1 Introduction . . . 3
1.2 Usage . . . 4
1.3 API Reference . . . 6
1.4 Contributing . . . 8
1.5 Changelog . . . 11

**2 Indices and tables** **13**

**Index** **15**

**django-anon** will help you anonymize your production database so it can be shared among developers, helping to reproduce bugs and make performance improvements in a production-like environment.

| | |
|---|---|
| | **Really fast** data anonymization and database operations using bulk updates to operate over huge tables |
| | **Flexible** to use your own anonymization functions or external libraries like Faker |
| | **Elegant** solution following consolidated patterns from projects like Django and Factory Boy |
| | **Powerful**. It can be used on any projects, not only Django, not only Python. Really! |

CHAPTER 1

# Table of Contents

## 1.1 Introduction

**django-anon** will help you anonymize your production database so it can be shared among developers, helping to reproduce bugs and make performance improvements in a production-like environment.

### 1.1.1 Features

| | |
|---|---|
| | **Really fast** data anonymization and database operations using bulk updates to operate over huge tables |
| | **Flexible** to use your own anonymization functions or external libraries like Faker |
| | **Elegant** solution following consolidated patterns from projects like Django and Factory Boy |
| | **Powerful**. It can be used on any projects, not only Django, not only Python. Really! |

### 1.1.2 Installation

```
pip install django-anon
```

### 1.1.3 Supported versions

- Python (2.7, 3.7)
- Django (1.8, 1.11, 2.2, 3.0)

### 1.1.4 License

MIT

## 1.2 Usage

Use `anon.BaseAnonymizer` to define your anonymizer classes:

```
import anon

from your_app.models import Person

class PersonAnonymizer(anon.BaseAnonymizer):
   email = anon.fake_email

   # You can use static values instead of callables
   is_admin = False

   class Meta:
      model = Person

# run anonymizer: be cautious, this will affect your current database!
PersonAnonymizer().run()
```

### 1.2.1 Built-in functions

```
import anon

anon.fake_word(min_size=_min_word_size, max_size=20)
anon.fake_text(max_size=255, max_diff_allowed=5, separator=' ')
anon.fake_small_text(max_size=50)
anon.fake_name(max_size=15)
anon.fake_username(max_size=10, separator='')
anon.fake_email(max_size=25, suffix='@example.com')
anon.fake_url(max_size=50, scheme='http://', suffix='.com')
anon.fake_phone_number(format='999-999-9999')
```

### 1.2.2 Lazy attributes

Lazy attributes can be defined as inline lambdas or methods, as shown below, using the `anon.lazy_attribute` function/decorator.

```
import anon

from your_app.models import Person

class PersonAnonymizer(anon.BaseAnonymizer):
   name = anon.lazy_attribute(lambda o: 'x' * len(o.name))

   @lazy_attribute
   def date_of_birth(self):
      # keep year and month
      return self.date_of_birth.replace(day=1)

   class Meta:
      model = Person
```

### 1.2.3 The clean method

```
import anon

class UserAnonymizer(anon.BaseAnonymizer):
   class Meta:
      model = User

   def clean(self, obj):
      obj.set_password('test')
      obj.save()
```

### 1.2.4 Defining a custom QuerySet

A custom QuerySet can be used to select the rows that should be anonymized:

```
import anon

from your_app.models import Person

class PersonAnonymizer(anon.BaseAnonymizer):
   email = anon.fake_email

   class Meta:
      model = Person

   def get_queryset(self):
      # keep admins unmodified
      return Person.objects.exclude(is_admin=True)
```

### 1.2.5 High-quality fake data

In order to be really fast, **django-anon** uses it's own algorithm to generate fake data. It is really fast, but the generated data is not pretty. If you need something prettier in terms of data, we suggest using Faker, which can be used out-of-the-box as the below:

```
import anon

from faker import Faker
from your_app.models import Address

faker = Faker()

class PersonAnonymizer(anon.BaseAnonymizer):
   postalcode = faker.postalcode

   class Meta:
      model = Address
```

## 1.3 API Reference

### 1.3.1 anon.BaseAnonymizer

**class** anon.**BaseAnonymizer**

**clean**(*obj*)
Use this function if you need to update additional data that may rely on multiple fields, or if you need to update multiple fields at once

**get_declarations**()
Returns ordered declarations. Any non-ordered declarations, for example any types that does not inherit from OrderedDeclaration will come first, as they are considered "raw" values and should not be affected by the order of other non-ordered declarations

**get_queryset**()
Override this if you want to delimit the objects that should be affected by anonymization

**patch_object**(*obj*)
Update object attributes with fake data provided by replacers

### 1.3.2 anon.lazy_attribute

anon.**lazy_attribute**(*lazy_fn*)
Returns LazyAttribute objects, that basically marks functions that should take *obj* as first parameter. This is useful when you need to take in consideration other values of *obj*

Example:

```
>>> full_name = lazy_attribute(o: o.first_name + o.last_name)
```

### 1.3.3 anon.fake_word

anon.**fake_word**(*min_size=1*, *max_size=20*)
Return fake word

**Min_size** Minimum number of chars

**Max_size** Maximum number of chars

Example:

```
>>> import django_anon as anon
>>> print(anon.fake_word())
adipisci
```

### 1.3.4 anon.fake_text

anon.**fake_text**(*max_size=255*, *max_diff_allowed=5*, *separator=' '*)
Return fake text

**Max_size** Maximum number of chars

**Max_diff_allowed** Maximum difference (fidelity) allowed, in chars number

**Separator** Word separator

Example:

```
>>> print(anon.fake_text())
alias aliquam aliquid amet animi aperiam architecto asperiores aspernatur␣
↪assumenda at atque aut autem beatae blanditiis commodi consectetur consequatur␣
↪consequuntur corporis corrupti culpa cum cumque cupiditate debitis delectus␣
↪deleniti deserunt dicta
```

### 1.3.5 anon.fake_small_text

`anon.`**`fake_small_text`**(*max_size=50*)

Preset for fake_text.

**Max_size** Maximum number of chars

Example:

```
>>> print(anon.fake_small_text())
Distinctio Dolor Dolore Dolorem Doloremque Dolores
```

### 1.3.6 anon.fake_name

`anon.`**`fake_name`**(*max_size=15*)

Preset for fake_text. Also returns capitalized words.

**Max_size** Maximum number of chars

Example:

```
>>> print(anon.fake_name())
Doloribus Ea
```

### 1.3.7 anon.fake_username

`anon.`**`fake_username`**(*max_size=10*, *separator=''*)

Returns fake username

**Max_size** Maximum number of chars

**Separator** Word separator

**Rand_range** Range to use when generating random number

Example:

```
>>> print(anon.fake_username())
eius54455
```

### 1.3.8 anon.fake_email

`anon.`**`fake_email`**(*max_size=40*, *suffix='@example.com'*)

Returns fake email address

**Max_size** Maximum number of chars

**Suffix** Suffix to add to email addresses (including @)

Example:

```
>>> print(anon.fake_email())
enim120238@example.com
```

### 1.3.9 anon.fake_url

`anon.fake_url`(*max_size=50*, *scheme='http://'*, *suffix='.com'*)

Returns fake URL

**Max_size** Maximum number of chars

**Scheme** URL scheme (http://)

**Suffix** Suffix to add to domain (including dot)

Example:

```
>>> print(anon.fake_url())
http://facilis.fuga.fugiat.fugit.harum.hic.id.com
```

### 1.3.10 anon.fake_phone_number

`anon.fake_phone_number`(*format='999-999-9999'*)

Returns a fake phone number in the desired format

**Format** Format of phone number to generate

Example:

```
>>> print(anon.fake_phone_number())
863-068-9424
```

## 1.4 Contributing

As an open source project, **django-anon** welcomes contributions of many forms.

Examples of contributions include:

- Code patches
- Documentation improvements
- Bug reports and code reviews

### 1.4.1 Code of conduct

Please keep the tone polite & professional. First impressions count, so let's try to make everyone feel welcome.

Be mindful in the language you choose. As an example, in an environment that is heavily male-dominated, posts that start 'Hey guys,' can come across as unintentionally exclusive. It's just as easy, and more inclusive to use gender neutral language in those situations. (e.g. 'Hey folks,')

The Django code of conduct gives a fuller set of guidelines for participating in community forums.

### 1.4.2 Issues

Some tips on good issue reporting:

- When describing issues try to phrase your ticket in terms of the behavior you think needs changing rather than the code you think need changing.
- Search the issue list first for related items, and make sure you're running the latest version of **django-anon** before reporting an issue.
- If reporting a bug, then try to include a pull request with a failing test case. This will help us quickly identify if there is a valid issue, and make sure that it gets fixed more quickly if there is one.
- Closing an issue doesn't necessarily mean the end of a discussion. If you believe your issue has been closed incorrectly, explain why and we'll consider if it needs to be reopened.

### 1.4.3 Development

To start developing on **django-anon**, clone the repo:

```
git clone https://github.com/Tesorio/django-anon
```

Changes should broadly follow the PEP 8 style conventions, and we recommend you set up your editor to automatically indicate non-conforming styles.

### 1.4.4 Coding Style

The Black code style is used across the whole codebase. Ideally, you should configure your editor to auto format the code. This means you can use **88 characters per line**, rather than 79 as defined by PEP 8.

Use *isort* to automate import sorting using the guidelines below:

- Put imports in these groups: future, stdlib, deps, local
- Sort lines in each group alphabetically by the full module name
- On each line, alphabetize the items with the upper case items grouped before the lowercase items

Don't be afraid, all specifications for linters are defined in `pyproject.toml` and `.flake8`

### 1.4.5 Testing

To run the tests, clone the repository, and then:

```
# Setup the virtual environment
python3 -m venv env
source env/bin/activate
pip install django
pip install -r tests/requirements.txt

# Run the tests
./runtests.py
```

### 1.4.6 Running against multiple environments

You can also use the excellent tox testing tool to run the tests against all supported versions of Python and Django. Install tox globally, and then simply run:

```
tox
```

### 1.4.7 Using pre-commit hook

CI will perform some checks during the build, but to save time, most of the checks can be ran locally beforing pushing code. To do this, we use pre-commit hooks. All you need to do, is to install and configure pre-commit:

```
pre-commit install --hook-type pre-push -f
```

### 1.4.8 Pull requests

It's a good idea to make pull requests early on. A pull request represents the start of a discussion, and doesn't necessarily need to be the final, finished submission.

It's also always best to make a new branch before starting work on a pull request. This means that you'll be able to later switch back to working on another separate issue without interfering with an ongoing pull requests.

It's also useful to remember that if you have an outstanding pull request then pushing new commits to your GitHub repo will also automatically update the pull requests.

GitHub's documentation for working on pull requests is available here.

Always run the tests before submitting pull requests, and ideally run tox in order to check that your modifications are compatible on all supported versions of Python and Django.

Once you've made a pull request take a look at GitHub Checks and make sure the tests are running as you'd expect.

### 1.4.9 Documentation

**django-anon** uses the Sphinx documentation system and is built from the `.rst` source files in the `docs/` directory.

To build the documentation locally, install Sphinx:

```
pip install Sphinx
```

Then from the `docs/` directory, build the HTML:

```
make html
```

To get started contributing, you'll want to read the reStructuredText reference.

### 1.4.10 Language style

Documentation should be in American English. The tone of the documentation is very important - try to stick to a simple, plain, objective and well-balanced style where possible.

Some other tips:

- Keep paragraphs reasonably short.

- Don't use abbreviations such as 'e.g.' but instead use the long form, such as 'For example'.

### 1.4.11 Releasing a new version

1. Bump the version in `anon/__init__.py`
2. Update the `CHANGELOG` file, moving items up from master to the new version
3. Submit a PR and wait for it to get approved/merged
4. Checkout to the corresponding commit and create a new tag: `python setup.py tag`
5. Build documentation for the new version (Requires access to ReadtheDocs.org)
6. Publish: `python setup.py publish` (Requires access to PyPI)

### 1.4.12 References

- https://github.com/encode/django-rest-framework/blob/master/CONTRIBUTING.md
- https://docs.djangoproject.com/en/dev/internals/contributing/

## 1.5 Changelog

**django-anon**'s release numbering works as follows:

- Versions are numbered in the form **A.B** or **A.B.C.**
- **A.B** is the feature release version number. Each version will be mostly backwards compatible with the previous release. Exceptions to this rule will be listed in the release notes.
- **C** is the patch release version number, which is incremented for bugfix and security releases. These releases will be 100% backwards-compatible with the previous patch release.

### 1.5.1 Releases


- *master*
- *0.3*
- *0.2*
- *0.1*


#### master

- ...

### 0.3

- Updated bulk_update method to use Django's built-in method if available
- Changed default `max_size` for `fake_email` to `40`
- Fixed error in `fake_text` when `max_size` is too short

### 0.2

- Added test for Django 3 using Python 3.7 in tox.ini
- Improved performance of fake_text
- Improved performance of BaseAnonymizer.patch_object
- Fix bug with get_queryset not being treated as reserved name
- Improved performance of fake_username
- Removed rand_range argument from fake_username (backwards incompatible)
- Changed select_chunk_size and update_batch_size to saner defaults

### 0.1

- Initial release

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Index


## B

`BaseAnonymizer` (*class in anon*), 6

## C

`clean()` (*anon.BaseAnonymizer method*), 6

## F

`fake_email()` (*in module anon*), 7
`fake_name()` (*in module anon*), 7
`fake_phone_number()` (*in module anon*), 8
`fake_small_text()` (*in module anon*), 7
`fake_text()` (*in module anon*), 6
`fake_url()` (*in module anon*), 8
`fake_username()` (*in module anon*), 7
`fake_word()` (*in module anon*), 6

## G

`get_declarations()` (*anon.BaseAnonymizer method*), 6
`get_queryset()` (*anon.BaseAnonymizer method*), 6

## L

`lazy_attribute()` (*in module anon*), 6

## P

`patch_object()` (*anon.BaseAnonymizer method*), 6