django-anon Documentation

Tesorio

Nov 12, 2019

Contents

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 Installation

```
pip install django-anon
```

1.2 Requirements

- Python (2.7, 3.6)
- Django (1.8, 1.11, 2.2)

1.3 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()
```

1.3.1 Built-in functions

```
fake_word(min_size=_min_word_size, max_size=20)
fake_text(max_size=255, max_diff_allowed=5, separator=' ')
fake_small_text(max_size=50)
fake_name(max_size=15)
fake_username(max_size=10, separator='', rand_range=(1000, 999999))
fake_email(max_size=25, suffix='@example.com')
fake_url(max_size=50, scheme='http://', suffix='.com')
fake_phone_number(format='999-999-9999')
```

1.3.2 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)
```

(continues on next page)

(continued from previous page)

```
   class Meta:
      model = Person
```

1.3.3 The clean method

```
import anon

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

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

1.3.4 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)
```

1.3.5 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
```

1.4 Changelog

Check out CHANGELOG.rst for release notes

1.5 Contributing

Check out CONTRIBUTING.rst for information about getting involved

1.6 License

MIT

Icon made by Eucalyp from www.flaticon.com

1.7 Contents

1.7.1 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.

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!

Table of Contents

1.7.2 Installation

```
pip install django-anon
```

1.7.3 Requirements

- Python (2.7, 3.6)
- Django (1.8, 1.11, 2.2)

1.7.4 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()
```

```
fake_word(min_size=_min_word_size, max_size=20)
fake_text(max_size=255, max_diff_allowed=5, separator=' ')
fake_small_text(max_size=50)
fake_name(max_size=15)
fake_username(max_size=10, separator='', rand_range=(1000, 999999))
fake_email(max_size=25, suffix='@example.com')
fake_url(max_size=50, scheme='http://', suffix='.com')
fake_phone_number(format='999-999-9999')
```

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
```

```
import anon

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

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

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)
```

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
```

1.7.5 Changelog

Check out CHANGELOG.rst for release notes

1.7.6 Contributing

Check out CONTRIBUTING.rst for information about getting involved

1.7.7 License

MIT

Icon made by Eucalyp from www.flaticon.com

1.7.8 Fake data functions

fake_word

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

fake_text

```
>>> 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
```

fake_small_text

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

fake_name

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

fake_username

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

fake_email

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

fake_url

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

fake_phone_number

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

1.7.9 Reference

BaseAnonymizer

lazy_attribute

Utils

1.7.10 Changelog

0.1

First version of django-anon

1.8 Indices and tables

- genindex
- modindex
- search