Django Sample Data Helper Documentation

Release 0.2.0

Jesús Espino García

August 07, 2016

Contents

CHAPTER 1

Install and configure

Install using pip, including any pillow if you want image generation...:

```
pip install django-sampledatahelper
pip install pillow  # For image generation
```

You should add the application to your django apps:

```
INSTALED_APPS += ["sampledatahelper"]
```

You can configure, if you want a `SAMPLEDATAHELPER_SEED` variable in your settings, to generate always the same data. Example:

```
SAMPLEDATAHELPER_SEED = 123456789
```

If you want to use the `sampledatafiller` command, you have to define your `SAMPLEDATAHELPER_MODELS` with the list of models you want to fill. Example:

```
SAMPLEDATAHELPER_MODELS = [
    # Generate 5 instances completely random
    { 'model': 'myapp.MyModel', 'number': 5, },

    # Generate 5 instances selecting random method for some fields
    {
        'model': 'myapp.MyModel',
        'number': 5,
        'fields_overwrite': [
            ('my_int_field', lambda _, sd: sd.int(5, 10)),
        ]
    },

    # Generate 5 instances with fixed data in a field
    {
        'model': 'myapp.MyModel',
        'number': 5,
        'fields_overwrite': [
            ('my_int_field', 5),
        ]
    }
]
```

CHAPTER 2

Quick start

Follow the install and configure instructions.

With Django sampledatahelper you have 2 options to populate your database

2.1 Using SampleDataFiller

Sample data filler is a command that use the `SAMPLEDATAHELPER_MODELS` setting variable to populate your database. Example:

```
SAMPLEDATAHELPER_MODELS = [
    # Generate 5 instances completly random
    { 'model': 'myapp.MyModel', 'number': 5, },

    # Generate 5 instances selecting random method for some fields
    {
        'model': 'myapp.MyModel',
        'number': 5,
        'fields_overwrite': [
            ('my_int_field', lambda _, sd: sd.int(5, 10)),
        ]
    },

    # Generate 5 instances with fixed data in a field
    {
        'model': 'myapp.MyModel',
        'number': 5,
        'fields_overwrite': [
            ('my_int_field', 5),
        ]
    }
]
```

Then you only have to run:

```
python manage.py sampledatafiller
```

2.2 Using a custom sampledata command

You can create a command to fill your models manullay to take more control.

If you have some aplications to populate, you can split your sample data generation on one command per app, or add only one command in one app thats generate everything.

The file must be in `<app-module>/management/commands/<command-name>.py` can be something like `myapp/management/commands/mysampledata.py`.

The easy way to build your command is using *ModelDataHelper*:

```
from django.core.management.base import BaseCommand
from myapp.models import MyModel
from sampledatahelper.model_helper import ModelDataHelper
from sampledatahelper.helper import SampleDataHelper

class Command(BaseCommand):
    args = ''
    help = 'Example data generator'
    mdh = ModelDataHelper(seed=12345678901)

    def handle(self, *args, **options):
        print "Generating MyModel data"
        # Generate 5 instances completly random
        self.mdh.fill_model(MyModel, 5)

        # Generate 5 instances selecting random method for some fields
        self.mdh.fill_model(MyModel,
                            5,
                            my_int_field=lambda instance, sd: sd.int(5, 10))

        # Generate 5 instances with fixed data in a field
        self.mdh.fill_model(MyModel, 5, my_int_field=8)
```

You can build a more precise command using directly the *SampleDataHelper*:

```
from django.core.management.base import BaseCommand
from myapp.models import MyModel
from sampledatahelper.helper import SampleDataHelper

class Command(BaseCommand):
    args = ''
    help = 'Example data generator'
    sd = SampleDataHelper(seed=12345678901)

    def generate_mymodel_data(self, instances):
        for x in range(instances):
            instance = MyModel.objects.create(
                slug=self.sd.slug(2, 3),
                name=self.sd.name(2, 3),
                claim=self.sd.sentence(),
                description=self.sd.paragraph(),
                email=self.sd.email(),
                photo=self.sd.image(64, 64),
                is_active=self.sd.boolean(),
                birth_date=self.sd.past_date(),
                expected_death_date=self.sd.future_date(),
                my_related_object=self.sd.db_object(MyRelatedModel)
            )

    def handle(self, *args, **options):
        print "Generating MyModel data"
        self.generate_mymodel_data(5)
```

To generate your sampledata, simply run the created command, for example:

```
python manage.py mysampledata
```

CHAPTER 3

Model Data Helper

Model data helper easy the models population introspecting in the django model fields.

class ModelDataHelper(*seed=None*)

Initialize the seed of the instance of model data helper, to allwais generate the same data.

ModelDataHelper.**fill_model**(*model*, *number*, **args*, ***kwargs*)

Generate a number of instances of the model and save it. You can overwrite the default data generator adding extra kwargs arguments.

To overwrite a field generation behavior you have to add extra arguments. with the name of the field, and the value must be, a fixed value or a callable object that receive 2 parameters, the model instance, and a SampleDataHelper instance. This overwrite is done alwais at the end of fill_model, this mean you can access all auto-generated data in other instance fields. This extra arguments can be a named argument using the field name as argument name, and the callable as value, or a not named name with value a tuple of field name and the callable. Examples:

```
fill_model(ModelName, 10, ('field_name', lambda instance, sd: sd.int()))
fill_model(ModelName, 10, field_name=lambda instance, sd: sd.int())
```

The order of field generation is, first the not overwrited fields in any order, second the overwrited fields in args, in the same order the parameters, and third the overwrited fields in kwargs in any orders. If you want to asure the ordering, use the args overwrite.

ModelDataHelper.**fill_model_instance**(*instance*, **args*, ***kwargs*)

Fill a instance of a django model. You can overwrite the default data generator adding extra arguments like in *fill_model* method. Examples:

```
fill_model_instance(instance, ('field_name', lambda instance, sd: sd.int()))
fill_model_instance(instance, field_name=lambda instance, sd: sd.int())
```

CHAPTER 4

SampleDataHelper

class `SampleDataHelper`(*seed=None*)
SampleDataHelper easy the random data generation for a lot of common used data types.

4.1 Number methods

`SampleDataHelper.int`(*min_value=0*, *max_value=sys.maxsize*)
Return an integer between min_value and max_value

`SampleDataHelper.number`(*ndigits*)
Return a number of n digits as max

`SampleDataHelper.digits`(*ndigits*)
Return a number of exactly n digits

`SampleDataHelper.float`(*min*, *max*)
Return a float from min to max

`SampleDataHelper.number_string`(*ndigits*)
Return a string of n digits

4.2 Text methods

`SampleDataHelper.char`()
Return a character between A-Z and a-z

`SampleDataHelper.chars`(*min_chars=1*, *max_chars=5*)
Return a string with n characters between A-Z and a-z being min_chars <= n <= max_chars

`SampleDataHelper.word`()
Returns a lorem ipsum word

`SampleDataHelper.words`(*min_words=1*, *max_words=5*)
Return a string with n lorem ipsum words being min_words <= n <= max_words

`SampleDataHelper.email`()
Return an email

`SampleDataHelper.url`()
Return an url

SampleDataHelper.**sentence**()
Return a lorem ipsum sentence (limited to 255 caracters)

SampleDataHelper.**short_sentence**()
Return a lorem ipsum sentence (limited to 100 caracters)

SampleDataHelper.**long_sentence**()
Return a lorem ipsum sentence (with 150 caracters or more)

SampleDataHelper.**paragraph**()
Return a lorem ipsum paragraph

SampleDataHelper.**paragraphs**(*min_paragraphs=1*, *max_paragraphs=5*)
Return a lorem ipsum text with n paragraphs being min_paragraphs <= n <= max_paragraphs

SampleDataHelper.**slug**(*min_words=5*, *max_words=5*)
Return a lorem ipsum slug between with n words being min_words <= n <= max_words

SampleDataHelper.**tags**(*min_tags=1*, *max_tags=5*, *tags_list=None*)
Return a string of n tags_list or lorem ipsum tags separated by commas being n max min_tags <= n <= max_tags

4.3 Time methods

SampleDataHelper.**date**(*begin=-365*, *end=365*)
Return a date between now+begin and now+end in days

SampleDataHelper.**date_between**(*min_date*, *max_date*)
Return a date between the min_date and max_date date objects

SampleDataHelper.**future_date**(*min_distance=0*, *max_distance=365*)
Return a future date between now+min_distance and now+max_distance in days

SampleDataHelper.**past_date**(*min_distance=0*, *max_distance=365*)
Return a past date between now-max_distance and now-min_distance in days

SampleDataHelper.**datetime**(*begin=-1440*, *end=1440*)
Return a datetime between now+begin and now+end in minutes

SampleDataHelper.**datetime_between**(*min_datetime*, *max_datetime*)
Return a datetime between the min_datetime and max_datetime datetime objects

SampleDataHelper.**future_datetime**(*min_distance=0*, *max_distance=1440*)
Return a future datetime between now+min_distance and now+max_distance in minutes

SampleDataHelper.**past_datetime**(*min_distance=0*, *max_distance=1440*)
Return a past datetime between now-max_distance and now-min_distance in minutes

SampleDataHelper.**time**()
Return a time

4.4 Localized methods

SampleDataHelper.**name**(*locale=None*, *number=1*, *as_list=False*)
Return a string or list of tipical names from locale using n names (compound names)

Supported locales: cat, es, fr, us

SampleDataHelper.**surname**(*locale=None*, *number=1*, *as_list=False*)
Return a string or list of tipical surnames from locale using n surnames

Supported locales: cat, es, fr, us

SampleDataHelper.**fullname**(*locale=None*, *as_list=False*)
Return a string or list of tipical names+surnames from locale

Supported locales: cat, es, fr, us

SampleDataHelper.**phone**(*locale*, *country_code*)
Return a phone number from a country with or without country code

Supported locales: es

SampleDataHelper.**zip_code**(*locale*)
Return a zip code for a country

Supported locales: es

SampleDataHelper.**state_code**(*locale*)
Return a state code for the locale country.

Supported locales: es, us

SampleDataHelper.**id_card**(*locale*)
Return a identification card code for a country

Supported locales: es

4.5 Image methods

SampleDataHelper.**image**(*width*, *height*, *typ="simple"*)
Return an image of width x height size generated with the typ generator.

Available typ generators: simple, plasma, mandelbrot, ifs, random

SampleDataHelper.**image_from_directory**(*directory_path, valid_extensions=['.jpg', '.bmp', '.png']*)
Return an image from a directory with a valid extension

4.6 Other methods

SampleDataHelper.**boolean**()
Return a boolean value

SampleDataHelper.**nullboolean**()
Return a boolean value or a None

SampleDataHelper.**ipv4**()
Return a ipv4 address

SampleDataHelper.**ipv6**()
Return a ipv6 address

SampleDataHelper.**mac_address**()
Return a mac address

SampleDataHelper.**hex_chars**(*min_chars=1*, *max_chars=5*)
Return a string with n characters between a-f and 0-9 being min_chars <= n <= max_chars

SampleDataHelper.**path**(*absolute=None*, *extension=''*, *min_levels=1*, *max_levels=5*)
Return a absolute or relative path (based on *absolute* parameter) string finished in *extension*, and with n levels being min_levels <= n <= max_levels

SampleDataHelper.**choice**(*choices*)
Return a value from a list

SampleDataHelper.**choices_key**(*choices*)
Return a key from a django choices list

SampleDataHelper.**db_object**(*model*, *raise_not_choices=True*)
Return a random object from the model. If no object found and raise_not_choices is True raises NotChoicesException.

The model may also be specified as a string in the form 'app_label.model_name'.

SampleDataHelper.**db_object_from_queryset**(*queryset*, *raise_not_choices=True*)
Return a random object from the queryset. If no object found and raise_not_choices is True raises NotChoicesException.

CHAPTER 5

Indices and tables

- genindex
- search

Z