

---

# **faker-file Documentation**

***Release 0.1***

**Artur Barseghyan <artur.barseghyan@gmail.com>**

**Feb 10, 2023**



# CONTENTS

<b>1</b>	<b>Prerequisites</b>	<b>3</b>
<b>2</b>	<b>Documentation</b>	<b>5</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Supported file types</b>	<b>9</b>
<b>5</b>	<b>Usage examples</b>	<b>11</b>
5.1	Standalone . . . . .	11
5.2	With Faker . . . . .	11
5.3	With factory_boy . . . . .	11
5.3.1	upload/models.py . . . . .	11
5.3.2	upload/factory.py . . . . .	12
<b>6</b>	<b>Testing</b>	<b>13</b>
<b>7</b>	<b>Writing documentation</b>	<b>15</b>
<b>8</b>	<b>License</b>	<b>17</b>
<b>9</b>	<b>Support</b>	<b>19</b>
<b>10</b>	<b>Author</b>	<b>21</b>
<b>11</b>	<b>Project documentation</b>	<b>23</b>
11.1	Quick start . . . . .	24
11.1.1	Installation . . . . .	24
11.1.2	Usage . . . . .	24
11.1.2.1	With Faker . . . . .	24
11.1.2.2	With factory_boy . . . . .	25
11.1.2.2.1	upload/models.py . . . . .	25
11.1.2.2.2	upload/factories.py . . . . .	25
11.2	Recipes . . . . .	26
11.2.1	When using standalone . . . . .	26
11.2.1.1	Prerequisites . . . . .	26
11.2.1.1.1	Create a TXT file with static content . . . . .	26
11.2.1.1.2	Create a DOCX file with dynamically generated content . . . . .	26
11.2.1.1.3	Create a ZIP file consisting of TXT files with static content . . . . .	27
11.2.1.1.4	Create a ZIP file consisting of 3 DOCX files with dynamically generated content . . . . .	27

11.2.2	When using with Faker . . . . .	27
11.2.2.1	Create a TXT file with static content . . . . .	28
11.2.2.2	Create a DOCX file with dynamically generated content . . . . .	28
11.2.3	When using with Django . . . . .	28
11.3	Release history and notes . . . . .	30
11.3.1	0.2 . . . . .	30
11.3.2	0.1 . . . . .	30
11.4	Package . . . . .	31
<b>12</b>	<b>Indices and tables</b>	<b>33</b>

**Generate fake files**



## **PREREQUISITES**

- Core package requires Python 3.7, 3.8, 3.9, 3.10 and 3.11.
- Django integration (with `factory_boy`) has been tested with Django 2.2, 3.0, 3.1, 3.2, 4.0 and 4.1.
- DOCX file support requires `python-docx`.
- ICO, JPEG, PNG, SVG and WEBP files support requires `imgkit`.
- PDF file support requires `pdftk`.
- PPTX file support requires `python-pptx`.





## DOCUMENTATION

Documentation is available on [Read the Docs](#).



## INSTALLATION

Latest stable version on PyPI:

```
pip install faker-file[all]
```

Or development version from GitHub:

```
pip install https://github.com/barseghyanartur/faker-file/archive/main.tar.gz
```



## **SUPPORTED FILE TYPES**

- DOCX
- ICO
- JPEG
- PDF
- PNG
- PPTX
- SVG
- TXT
- WEBP
- ZIP



## USAGE EXAMPLES

### 5.1 Standalone

```
from faker_file.providers.txt_file import TxtFileProvider

file = TxtFileProvider(None).txt_file()
```

### 5.2 With Faker

```
from faker import Faker
from faker_file.providers.txt_file import TxtFileProvider

FAKER = Faker()
FAKER.add_provider(TxtFileProvider)

file = FAKER.txt_file()
```

### 5.3 With factory\_boy

#### 5.3.1 upload/models.py

```
from django.db import models

class Upload(models.Model):

    # ...
    upload = models.FileField()
```

### 5.3.2 upload/factory.py

Note, that when using `faker-file` with Django, you need to pass your `MEDIA_ROOT` setting as `root_path` value (which is by default set to `tempfile.gettempdir()`).

```
import factory
from django.conf import settings
from factory import Faker
from factory.django import DjangoModelFactory
from faker_file.providers.docx_file import DocxFileProvider

from upload.models import Upload

factory.Faker.add_provider(DocxFileProvider)

class UploadFactory(DjangoModelFactory):

    # ...
    file = Faker("docx_file", root_path=settings.MEDIA_ROOT)

    class Meta:
        model = Upload
```



**TESTING**

Simply type:

```
pytest -vvv
```

Or use tox:

```
tox
```

Or use tox to check specific env:

```
tox -e py310-django41
```



## WRITING DOCUMENTATION

Keep the following hierarchy.

```
====  
title  
====  
  
header  
=====  
  
sub-header  
-----  
  
sub-sub-header  
~~~~~  
  
sub-sub-sub-header  
^^^^^^  
  
sub-sub-sub-sub-header  
+++++++  
  
sub-sub-sub-sub-sub-header  
*****
```



**LICENSE**

MIT



## SUPPORT

For any security issues contact me at the e-mail given in the *Author* section.

For overall issues, go to [GitHub](#).





---

CHAPTER  
TEN

---

AUTHOR

Artur Barseghyan <[artur.barseghyan@gmail.com](mailto:artur.barseghyan@gmail.com)>



## PROJECT DOCUMENTATION

Contents:

### Table of Contents

- *faker-file*
  - *Prerequisites*
  - *Documentation*
  - *Installation*
  - *Supported file types*
  - *Usage examples*
    - \* *Standalone*
    - \* *With Faker*
    - \* *With factory\_boy*
      - *upload/models.py*
      - *upload/factory.py*
  - *Testing*
  - *Writing documentation*
  - *License*
  - *Support*
  - *Author*
  - *Project documentation*
  - *Indices and tables*

## 11.1 Quick start

### 11.1.1 Installation

```
pip install faker-file[all]
```

### 11.1.2 Usage

#### 11.1.2.1 With Faker

```
from faker import Faker
from faker_file.providers.docx_file import DocxFileProvider
from faker_file.providers.ico_file import IcoFileProvider
from faker_file.providers.jpeg_file import JpegFileProvider
from faker_file.providers.pdf_file import PdfFileProvider
from faker_file.providers.png_file import PngFileProvider
from faker_file.providers.pptx_file import PptxFileProvider
from faker_file.providers.svg_file import SvgFileProvider
from faker_file.providers.txt_file import TxtFileProvider
from faker_file.providers.webp_file import WebpFileProvider
from faker_file.providers.zip_file import ZipFileProvider

FAKER = Faker()
FAKER.add_provider(DocxFileProvider)
FAKER.add_provider(IcoFileProvider)
FAKER.add_provider(JpegFileProvider)
FAKER.add_provider(PdfFileProvider)
FAKER.add_provider(PngFileProvider)
FAKER.add_provider(PptxFileProvider)
FAKER.add_provider(SvgFileProvider)
FAKER.add_provider(TxtFileProvider)
FAKER.add_provider(TxtFileProvider)
FAKER.add_provider(WebpFileProvider)

docx_file = FAKER.docx_file()
ico_file = FAKER.ico_file()
jpeg_file = FAKER.jpeg_file()
pdf_file = FAKER.pdf_file()
png_file = FAKER.png_file()
pptx_file = FAKER.pptx_file()
svg_file = FAKER.svg_file()
txt_file = FAKER.txt_file()
webp_file = FAKER.webp_file()
zip_file = FAKER.zip_file()
```

### 11.1.2.2 With factory\_boy

#### 11.1.2.2.1 upload/models.py

```
from django.db import models

class Upload(models.Model):
    """Upload model."""

    name = models.CharField(max_length=255, unique=True)
    description = models.TextField(null=True, blank=True)

    # Files
    docx_file = models.FileField()
    pdf_file = models.FileField()
    pptx_file = models.FileField()
    txt_file = models.FileField()
    zip_file = models.FileField()

    class Meta:
        verbose_name = "Upload"
        verbose_name_plural = "Upload"

    def __str__(self):
        return self.name
```

#### 11.1.2.2.2 upload/factories.py

```
from django.conf import settings

from factory import Faker
from factory.django import DjangoModelFactory

# Import all providers we want to use
from faker_file.providers.docx_file import DocxFileProvider
from faker_file.providers.pdf_file import PdfFileProvider
from faker_file.providers.pptx_file import PptxFileProvider
from faker_file.providers.txt_file import TxtFileProvider
from faker_file.providers.zip_file import ZipFileProvider

from upload.models import Upload

# Add all providers we want to use
Faker.add_provider(DocxFileProvider)
Faker.add_provider(PdfFileProvider)
Faker.add_provider(PptxFileProvider)
Faker.add_provider(TxtFileProvider)
Faker.add_provider(ZipFileProvider)

class UploadFactory(DjangoModelFactory):
```

(continues on next page)

(continued from previous page)

```

"""Upload factory."""

name = Faker("text", max_nb_chars=100)
description = Faker("text", max_nb_chars=1000)

# Files
docx_file = Faker("docx_file", root_path=settings.MEDIA_ROOT)
pdf_file = Faker("pdf_file", root_path=settings.MEDIA_ROOT)
pptx_file = Faker("pptx_file", root_path=settings.MEDIA_ROOT)
txt_file = Faker("txt_file", root_path=settings.MEDIA_ROOT)
zip_file = Faker("zip_file", root_path=settings.MEDIA_ROOT)

class Meta:
    model = Upload

```

## 11.2 Recipes

### 11.2.1 When using standalone

#### 11.2.1.1 Prerequisites

##### Imports

```

import Faker
from faker_file.providers.docx_file import DocxFileProvider
from faker_file.providers.pdf_file import PdfFileProvider
from faker_file.providers.pptx_file import PptxFileProvider
from faker_file.providers.txt_file import TxtFileProvider
from faker_file.providers.zip_file import ZipFileProvider

```

##### 11.2.1.1.1 Create a TXT file with static content

- Content of the file is Lorem ipsum.

```

file = TxtFileProvider(None).txt_file(content="Lorem ipsum")

```

##### 11.2.1.1.2 Create a DOCX file with dynamically generated content

- Content is generated dynamically.
- Content is limited to 1024 chars.
- Wrap lines after 80 chars.
- Prefix the filename with zzz.

```

file = DocxFileProvider(None).docx_file(
    max_nb_chars=1_024,

```

(continues on next page)

(continued from previous page)

```

wrap_chars_after=80,
prefix="zzz",
)

```

#### 11.2.1.1.3 Create a ZIP file consisting of TXT files with static content

- 5 TXT files in the ZIP archive (default value is 5).
- Content of all files is Lorem ipsum.

```
file = ZipFileProvider(None).zip_file(options={"content": "Lorem ipsum"})
```

#### 11.2.1.1.4 Create a ZIP file consisting of 3 DOCX files with dynamically generated content

- 3 DOCX files in the ZIP archive.
- Content is generated dynamically.
- Content is limited to 1024 chars.
- Prefix the filenames in archive with xxx\_.
- Prefix the filename of the archive itself with zzz.
- Inside the ZIP, put all files in directory yyy.

```

from faker_file.providers.zip_file import create_inner_docx_file
file = ZipFileProvider(None).zip_file(
    prefix="zzz",
    options={
        "count": 3,
        "create_inner_file_func": create_inner_docx_file,
        "max_nb_chars": 1_024,
        "prefix": "xxx_",
        "directory": "yyy",
    }
)

```

## 11.2.2 When using with Faker

### Imports and initialization

```

import Faker
from faker_file.providers.docx_file import DocxFileProvider
from faker_file.providers.pdf_file import PdfFileProvider
from faker_file.providers.pptx_file import PptxFileProvider
from faker_file.providers.txt_file import TxtFileProvider
from faker_file.providers.zip_file import ZipFileProvider

FAKER = Faker()
FAKER.add_provider(DocxFileProvider)

```

(continues on next page)

(continued from previous page)

```
FAKER.add_provider(PdfFileProvider)
FAKER.add_provider(PptxFileProvider)
FAKER.add_provider(TxtFileProvider)
FAKER.add_provider(ZipFileProvider)
```

### 11.2.2.1 Create a TXT file with static content

```
file = FAKER("txt_file", content="Lorem ipsum dolor sit amet")
```

### 11.2.2.2 Create a DOCX file with dynamically generated content

- Content is generated dynamically.
- Content is limited to 1024 chars.
- Wrap lines after 80 chars.
- Prefix the filename with zzz.

```
file = FAKER(
    "docx_file",
    max_nb_chars=1_024,
    wrap_chars_after=80,
    prefix="zzz",
)
```

## 11.2.3 When using with Django

When used with Django (to generate fake data with `factory_boy` factories), the `root_path` argument shall be provided. Otherwise (although no errors will be triggered) the generated files will reside outside the `MEDIA_ROOT` directory (by default in `/tmp/tmp/` on Linux) and further operations with those files through Django will cause `SuspiciousOperation` exception.

### Imaginary `Django` model

```
from django.db import models

class Upload(models.Model):
    """Upload model."""

    name = models.CharField(max_length=255, unique=True)
    description = models.TextField(null=True, blank=True)

    # Files
    docx_file = models.FileField()
    pdf_file = models.FileField()
    pptx_file = models.FileField()
    txt_file = models.FileField()
    zip_file = models.FileField()
    file = models.FileField()
```

(continues on next page)



(continued from previous page)

```

class Meta:
    verbose_name = "Upload"
    verbose_name_plural = "Upload"

def __str__(self):
    return self.name

```

Correspondent `factory\_boy` factory

```

from django.conf import settings

from factory import Faker
from factory.django import DjangoModelFactory

# Import all providers we want to use
from faker_file.providers.docx_file import DocxFileProvider
from faker_file.providers.pdf_file import PdfFileProvider
from faker_file.providers.pptx_file import PptxFileProvider
from faker_file.providers.txt_file import TxtFileProvider
from faker_file.providers.zip_file import ZipFileProvider

from upload.models import Upload

# Add all providers we want to use
Faker.add_provider(DocxFileProvider)
Faker.add_provider(PdfFileProvider)
Faker.add_provider(PptxFileProvider)
Faker.add_provider(TxtFileProvider)
Faker.add_provider(ZipFileProvider)

class UploadFactory(DjangoModelFactory):
    """Upload factory."""

    name = Faker("text", max_nb_chars=100)
    description = Faker("text", max_nb_chars=1000)

    # Files
    docx_file = Faker("docx_file", root_path=settings.MEDIA_ROOT)
    pdf_file = Faker("pdf_file", root_path=settings.MEDIA_ROOT)
    pptx_file = Faker("pptx_file", root_path=settings.MEDIA_ROOT)
    txt_file = Faker("txt_file", root_path=settings.MEDIA_ROOT)
    zip_file = Faker("zip_file", root_path=settings.MEDIA_ROOT)
    file = Faker("txt_file", root_path=settings.MEDIA_ROOT)

    class Meta:
        model = Upload

```

```

from random import choice

from factory import LazyAttribute

```

(continues on next page)

(continued from previous page)

```
PROVIDER_CHOICES = [  
    lambda: DocxFileProvider(None).docx_file(root_path=settings.MEDIA_ROOT),  
    lambda: PdfFileProvider(None).pdf_file(root_path=settings.MEDIA_ROOT),  
    lambda: PptxFileProvider(None).pptx_file(root_path=settings.MEDIA_ROOT),  
    lambda: TxtFileProvider(None).txt_file(root_path=settings.MEDIA_ROOT),  
    lambda: ZipFileProvider(None).zip_file(root_path=settings.MEDIA_ROOT),  
]  
  
def pick_random_provider(*args, **kwargs):  
    return choice(PROVIDER_CHOICES)()  
  
class UploadFactory(DjangoModelFactory):  
    """Upload factory that randomly picks a file provider."""  
  
    # ...  
    file = LazyAttribute(pick_random_provider)  
    # ...
```

## 11.3 Release history and notes

Sequence based identifiers are used for versioning (schema follows below):

```
major.minor[.revision]
```

- It's always safe to upgrade within the same minor version (for example, from 0.3 to 0.3.4).
- Minor version changes might be backwards incompatible. Read the release notes carefully before upgrading (for example, when upgrading from 0.3.4 to 0.4).
- All backwards incompatible changes are mentioned in this document.

### 11.3.1 0.2

2022-12-07

- Added support for ICO, JPEG, PNG, SVG and WEBP files.
- Documentation improvements.

### 11.3.2 0.1

2022-12-06

- Initial beta release.

## 11.4 Package

Contents:

Table of Contents
-------------------

- |  |
|--|
| <ul style="list-style-type: none"><li>• <i>Package</i></li></ul> |
|--|



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`