django-encode Documentation

Release 1.0.3

Collab

December 15, 2015

Contents

This is the documentation for django-encode 1.0.3, generated on December 15, 2015.

CHAPTER 1

Developer Guide

1.1 Models

1.2 Tasks

1.3 Encoders

Encoders.

get_encoder_class(*import_path=None*)
Get the encoder class by supplying a fully qualified path to `import_path`.

If `import_path` is `None` the default encoder class specified in the `ENCODE_DEFAULT_ENCODER_CLASS` is returned.

Parameters **import_path** (*str*) – Fully qualified path of the encoder class, for example: `encode.encoders.BasicEncoder`.

Returns The encoder class.

Return type class

class BaseEncoder(*profile*, *input_path=None*, *output_path=None*)
The base encoder.

Parameters

- **profile** (`EncodingProfile`) – The encoding profile that configures this encoder.
- **input_path** (*str*) –
- **output_path** (*str*) –

command
The command for the encoder with the vars injected, eg. `convert "/path/to/input.gif" "/path/to/output.png"`.

Return type str

Returns The command.

class BasicEncoder(*profile*, *input_path=None*, *output_path=None*)
Encoder that uses the `subprocess` module.

start()
Start encoding.

Raises `EncodeError` if something goes wrong during encoding.

class FFMpegEncoder(*profile*, *input_path=None*, *output_path=None*)
Encoder that uses the FFMpeg tool.

start()
Start encoding.

Raises `EncodeError` if something goes wrong during encoding.

1.4 Utilities

Utilities.

fqn(*obj*)
Get fully qualified name of `obj`, eg. `encode.util.fqn`.

Parameters **obj** –

Return type str

get_random_filename(*file_extension=u'png'*, *length=12*)
Returns a random filename with an optional length and file-extension.

Parameters

- **file_extension** (*str*) – File extension for the file, e.g. 'gif'.
- **length** (*int*) – Number of random characters the filename should contain.

Return type str

Returns A random filename, e.g. `4AwV8Ckn65a3.png`.

get_media_upload_to(*instance*, *filename*)
Get target path for user file uploads.

Parameters

- **instance** (`django.db.models.Model`) – Model instance.
- **filename** (*str*) – The filename for the file being uploaded, eg. 'test.png'.

Return type str

parseMedia(*data*)
Decode base64-encoded media data and return result.

Parameters **data** (*str*) – base64-encoded string

Return type str

storeMedia(*model*, *inputFileField*, *title*, *profiles*, *fpath*)
Encode and store `MediaBase` object.

Parameters

- **model** (*class*) – A model object or instance, e.g. `Video`.
- **title** (*str*) – Name of the file, e.g. *test.png*.
- **profiles** (*list*) – List of `EncodingProfile` names.

- **fpath** (*str*) – Location of media file.

Variables **inputFileField** – Name of the model field where the file will be stored.

Return type `MediaBase` subclass.

class **TemporaryMediaFile** (*prefix*, *model*, *inputFileField*, *profiles*, *extension=u'media'*)

Container to store a temporary media file for encoding.

Parameters

- **prefix** (*str*) – The prefix to use for the temporary filename, e.g. `video_`.
- **model** (`django.db.models.Model`) – The model to store the file on, e.g. a subclass of `MediaBase`.
- **inputFileField** (*str*) – Name of the model field where the file will be stored.
- **profiles** (*list*) – List of `EncodingProfile` names, e.g. `[u"MP4", u"WebM Audio/Video"]`
- **extension** (*str*) – The extension to use for the temporary filename. Defaults to `media`.

save (*fileData*)

Save `fileData` in temporary file and start encoding.

Parameters **fileData** (`io.BytesIO`) – The media bytes.

Return type `MediaBase`

Returns A new instance of type `self.model`.

1.5 Settings

Configuration options.

class **EncodeConf** (***kwargs*)

Configuration settings.

MEDIA_PATH_NAME = 'encode_test'

Name of the root directory holding the user-uploaded files.

MEDIA_ROOT = '/home/docs/checkouts/readthedocs.org/user_builds/django-encode/checkouts/latest/doc/media'

Absolute filesystem path to the directory that will hold user-uploaded files for the encode application.

AUDIO_PROFILES = ['MP3 Audio', 'Ogg Audio']

TODO

VIDEO_PROFILES = ['MP4', 'WebM Audio/Video']

TODO

IMAGE_PROFILES = ['PNG']

TODO

LOCAL_FILE_STORAGE = 'django.core.files.storage.FileSystemStorage'

TODO

REMOTE_FILE_STORAGE = 'django.core.files.storage.FileSystemStorage'

Django file storage used for transferring media uploads to the encoder.

CDN_FILE_STORAGE = 'django.core.files.storage.FileSystemStorage'

Django file storage used for storing encoded media on a CDN network.

LOCAL_STORAGE_OPTIONS = {'location': '/home/docs/checkouts/readthedocs.org/user_builds/django-encode/checkouts/
TODO

REMOTE_STORAGE_OPTIONS = {'location': '/home/docs/checkouts/readthedocs.org/user_builds/django-encode/checkout
TODO

DEFAULT_ENCODER_CLASS = 'encode.encoders.BasicEncoder'
TODO

1.6 Development

After checkout, install dependencies and package in active virtualenv:

```
$ pip install -r requirements/development.txt
$ pip install -r requirements/testing.txt
$ pip install -r requirements/production.txt
$ pip install -e .
```

1.6.1 Testing

Running tests with Tox:

```
$ tox -v
```

Or alternatively:

```
$ python setup.py test
```

Running tests without Tox:

```
$ ./runtests.py
```

Directly with *django-admin*:

```
$ django-admin test --settings=encode.tests.settings encode
```

1.6.2 Coverage

To generate a test coverage report using coverage.py:

```
$ coverage run --source='.' runtests.py
$ coverage html
```

The resulting HTML report can be found in the `htmlcov` directory.

1.6.3 Localization

To collect all strings for the locale `nl` into `django.po`:

```
$ django-admin makemessages --settings=encode.tests.settings --ignore=tests/*.py -l nl
```

After translating, compile the `django.po` catalog into the binary version *django.mo*:

```
$ django-admin compilemessages --settings=encode.tests.settings
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

e