django-herokuify Documentation

Release 1.0.pre3

Filip Wasilewski

January 22, 2013

CONTENTS

CHAPTER

ONE

QUICKSTART

Simplify Django configuration in two easy steps:

1. Include `django-herokuify` and `pylibmc` packages in your `requirements.txt` file.
2. In the Django `settings.py` of your Heroku project add:

```
import herokuify

from herokuify.common import *          # Common settings, SSL proxy header
from herokuify.aws import *             # AWS access keys
from herokuify.mail.mailgun import *    # Mailgun email add-on settings
from herokuify.mail.sendgrid import *   # ... or Sendgrid

DATABASES = herokuify.get_db_config()   # Database config
CACHES = herokuify.get_cache_config()   # Memcache config for Memcache/MemCachier
```

Additionally, you can also use storage backends that works nice with Amazon S3 and Django Compressor:

```
DEFAULT_FILE_STORAGE = "herokuify.storage.S3MediaStorage"
MEDIA_URL = "https://{0}.s3.amazonaws.com/media/".format(AWS_STORAGE_BUCKET_NAME)

STATICFILES_STORAGE = "herokuify.storage.CachedS3StaticStorage"
STATIC_URL = "https://{0}.s3.amazonaws.com/static/".format(AWS_STORAGE_BUCKET_NAME)

COMPRESS_STORAGE = "herokuify.storage.CachedS3StaticStorage"
COMPRESS_OFFLINE = True
```

CHAPTER

TWO

DEPENDENCIES

`django-herokuify` installs the following dependencies by default:

- Caching and cache configuration:

```
pylibmc>=1.2.3
django-pylibmc-sasl>=0.2.4
django-heroku-memcacheify>=0.3
```

- DB configuration:

```
dj-database-url>=0.2.1
django-heroku-postgresify>=0.2
```

- Storage backend:

```
django-storages>=1.1.5
boto>=2.6.0
```

Note: It is necessary include `pylibmc` package entry in your project's root `requirements.txt` file. The Heroku Django buildpack checks for this entry and configures the `libmemcached` C build dependency.

CHAPTER

THREE

CONTENT

3.1 Common settings

Commonly used configuration.

3.2 Database configuration

Provides `herokuify.db.get_db_config()` function that reads PostgreSQL database settings from Heroku environment.

Note: This functionality relies on django-heroku-postgresify.

3.2.1 Django config

Use this to configure database settings in your Django project:

```
import herokuify
DATABASES = herokuify.get_db_config()
```

3.3 Cache

Provides `herokuify.cache.get_cache_config()` that reads Memcache cache settings from Heroku environment for Memcache or MemCachier add-ons.

Note: This functionality relies on django-heroku-memcacheify.

3.3.1 Django config

Use this to configure cache settings in your Django project:

```
import herokuify
CACHES = herokuify.get_cache_config()
```

3.3.2 Heroku config

Use `heroku addons` command to subscribe for Memcache or MemCachier.

Quick start with either:

```
heroku addons:add memcache:5mb
```

```
heroku addons:add memcachier:dev
```

3.4 Email configuration

Note: For now the project includes only Mailgun and Sendgrid configuration. Feel free to contribute definitions for other providers.

3.4.1 Mailgun add-on

Retrieves Mailgun add-on settings from the environment and defines the following variables in the `herokuify.mail.mailgun` module:

- `MAILGUN_API_KEY`
- `MAILGUN_SMTP_SERVER`
- `MAILGUN_SMTP_LOGIN`
- `MAILGUN_SMTP_PASSWORD`
- `MAILGUN_SMTP_PORT`

as well as `EMAIL_*` settings for use in Django config:

- `EMAIL_HOST`
- `EMAIL_HOST_USER`
- `EMAIL_HOST_PASSWORD`
- `EMAIL_PORT`

Django config

Import settings from this module into your Django project settings:

```
from herokuify.mail.mailgun import EMAIL_HOST, EMAIL_HOST_USER, EMAIL_HOST_PASSWORD, EMAIL_PORT
```

Installing add-on

Use `heroku addons` command to subscribe for Mailgun add-on:

```
heroku addons:add mailgun:<PLAN>
```

Quick start with:

```
heroku addons:add mailgun:starter
```

3.4.2 Sendgrid add-on

Retrieves Sendgrid add-on settings from the environment and defines the following variables in the `herokuify.mail.sendgrid` module:

- `SENDGRID_SMTP_SERVER`
- `SENDGRID_USERNAME`
- `SENDGRID_PASSWORD`
- `SENDGRID_SMTP_PORT`

as well as `EMAIL_*` settings for use in Django config:

- `EMAIL_HOST`
- `EMAIL_HOST_USER`
- `EMAIL_HOST_PASSWORD`
- `EMAIL_PORT`
- `EMAIL_USE_TLS`

Django config

Import settings from this module into your Django project settings:

```
from herokuify.mail.sendgrid import EMAIL_HOST, EMAIL_HOST_USER, EMAIL_HOST_PASSWORD, EMAIL_PORT, EMA
```

Installing add-on

See Sendgrid add-on page. Quick start with:

```
heroku addons:add sendgrid:starter
```

3.4.3 Defined module attributes

Mailgun

Sendgrid

3.5 Amazon AWS settings

Reads common AWS settings like `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_STORAGE_BUCKET_NAME` from Heroku environment.

3.5.1 Django config

Simply import settings from this module into your Django project settings file:

```
from herokuify.aws import AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_STORAGE_BUCKET_NAME
```

3.5.2 Heroku config

Use `heroku config` command to define the values:

```
heroku config:add AWS_ACCESS_KEY_ID=<key id>
heroku config:add AWS_SECRET_ACCESS_KEY=<secret key>
heroku config:add AWS_STORAGE_BUCKET_NAME=<bucket name>
```

3.5.3 Defined model attributes

3.6 Storage Backends

Defines S3 bucket subdirectory storage for static and media files and provides storage backends that are compatible with Django Compressor.

3.6.1 Django config

To use AWS S3 storage for static and media files set this in your project settings:

```
STATICFILES_STORAGE = "herokuify.storage.S3StaticStorage"
STATIC_URL = "https://{0}.s3.amazonaws.com/static/".format(AWS_STORAGE_BUCKET_NAME)

DEFAULT_FILE_STORAGE = "herokuify.storage.S3MediaStorage"
MEDIA_URL = "https://{0}.s3.amazonaws.com/media/".format(AWS_STORAGE_BUCKET_NAME)
```

There's also a storage that works well with Django Compressor, collectstatic command and offline assets compression:

```
STATICFILES_STORAGE = "herokuify.storage.CachedS3StaticStorage"
STATIC_URL = "https://{0}.s3.amazonaws.com/static/".format(AWS_STORAGE_BUCKET_NAME)

COMPRESS_STORAGE = "herokuify.storage.CachedS3StaticStorage"
COMPRESS_OFFLINE = True
```

Remember to configure and include *Amazon AWS settings* config in your settings as well enable user environment variables in the [Slug compilation](https://devcenter.heroku.com/articles/slug-compiler) build phase, so the storage backend is able to connect to S3 when executing `collectstatic` and `compress` commands:

Note: Django on Heroku: installing NodeJS and Less for static assets compilation and Django and Heroku Cookbook provides more information and build scripts for automatic static files compression during deployment.

3.6.2 Available storage backends

S3 Static and Media storage

Django Compressor-compatible storage