django-common-configs Documentation

Release 0.1.0.dev1

Filip Wasilewski

January 23, 2014

Contents

CHAPTER 1

django-common-configs

Common Configuration settings for Django projects.

Goes in line with 12 factor app and popular hosting platforms like Heroku.

Developed and used at en.ig.ma software shop.

1.1 Overview

Getting Django and popular apps settings right require time and a bit of experience.

This project provides predefined and verified configs for various aspects of Django apps, promotes convention over configuration and allows to keep `settings.py` file DRY.

It covers security, static files, assets compression, storage, AWS, celery, Sentry, logging, integration with common services, Heroku and more.

Developed at en.ig.ma software shop and used in multiple projects.

1.2 Documentation

The full documentation is at http://django-common-configs.rtfd.org.

1.3 Quickstart

Simplify Django project configuration in two easy steps:

Include `django-common-configs` and other related packages in your `requirements.txt` file.

Install django-configurations, add required common config mixins (they are just plain Python classes) to your `settings.py` Configuration classes and override base settings as necessary:

```
from common_configs import Configuration, values

from common_configs.django import Locale, SingleSite, DjangoSecurity
from common_configs.apps import CrispyForms, Imagekit, CeleryDev, CompressDev, CompressProd
from common_configs.logging import StructLoggingDev, StructLoggingProd
from common_configs.paas.heroku import Heroku, CeleryHerokuBigWig
from common_configs.services import APNS, GCM, CacheDev, AWS, Mailgun, Sendgrid, Raven, Pusher, Twili
from common_configs.storage import LocalCompressStorage, AWSCompressStorage
```

```
class Common(Locale, SingleSite,
             CrispyForms, Imagekit, APNS, GCM,
             Configuration):

    DEBUG = False
    TEMPLATE_DEBUG = False


class DevConfig(LocalCompressStorage, CeleryDev, CompressDev, StructLoggingDev, CacheDev,
                Common):

    DEBUG = True
    TEMPLATE_DEBUG = True
    DATABASES = values.DatabaseURLValue("postgres://...")


class ProdConfig(AWS, AWSCompressStorage, CeleryHerokuBigWig, CompressProd, StructLoggingProd,
                 Heroku, Mailgun, Raven, Pusher, Twilio,
                 DjangoSecurity,
                 Common):
    pass
```

1.4 Dependencies

`django-common-configs` depends on `django-configurations>=0.7` and optionally on the following packages:

Module	Requirements
security	`django-secure>=1.0`, `django_csp>=2.0.3`
compress	`django_compressor>=1.3`
debug	`django-debug-toolbar>=1.0.1`
auth	`django-allauth>=0.15.0`
forms	`django-crispy-forms>=1.4.0`
imagekit	`django-imagekit>=3.2`
pusher	`pusher>=0.8`
sentry	`raven>=4.0.3`
storage	`boto>=2.23.0`, `django-storages>=1.1.8`, `Collectfast>=0.1.13`
logging	`django-log-request-id>=0.0.3`
structlog	`structlog>=0.4.1`, `django-log-request-id>=0.0.3`
twilio	`twilio`
heroku	`django-pylibmc-sasl>=0.2.4`, `django-heroku-memcacheify>=0.4`, `django-heroku-postgresify>=0.3`

All dependencies can be easily added to your `requirements.txt` file by specifying it using pip syntax:

```
django-common-configs[security,compress,debug,auth,forms,imagekit,pusher,sentry,storage,structlog,twi
```

1.5 License

`django-common-configs` is released under the BSD license.

1.6 Other Resources

- GitHub repository - https://github.com/nigma/django-common-configs
- PyPi Package site - http://pypi.python.org/pypi/django-common-configs

1.7 Content

1.7.1 Installation

Add `django-common-configs=<version>` to your `requirements.txt` file or install it directly from the command line by invoking:

```
$ pip install django-common-configs
```

or:

```
$ easy_install django-common-configs
```

Additional requirements can be easily installed by specifying list of modules that are going to be used. For example:

```
django-common-configs[security,compress,forms,imagekit,pusher,sentry,storage,structlog,twilio,heroku
```

See *Dependencies* for list of optional requirements for each module.

Note: Please note that this package is under development and it is recommended to specify an exact package version number in your requirements.txt file.

1.7.2 Django settings mixins

DjangoSecurity

Enable SSL and other security settings for Django and django-secure app.

Django-secure install instructions: http://django-secure.rtfd.org/latest/index.html#installation

DjangoSecurity

class `common_configs.django.security.`**`DjangoSecurity`**

Configures some good defaults for non-SSL sites.

For SSL-enabled sites use `DjangoSSLSecurity`.

`SESSION_COOKIE_SECURE` = **BooleanValue**

Env variable: DJANGO_SESSION_COOKIE_SECURE

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-SESSION_COOKIE_SECURE

SESSION_COOKIE_HTTPONLY = Value: True
Env variable: DJANGO_SESSION_COOKIE_HTTPONLY

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-SESSION_COOKIE_HTTPONLY

CSRF_COOKIE_SECURE = BooleanValue
Env variable: DJANGO_CSRF_COOKIE_SECURE

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-CSRF_COOKIE_SECURE

SECURE_FRAME_DENY = Value: True
Env variable: DJANGO_SECURE_FRAME_DENY

http://django-secure.rtfd.org/latest/settings.html#secure-frame-deny

SECURE_CONTENT_TYPE_NOSNIFF = Value: True
Env variable: DJANGO_SECURE_CONTENT_TYPE_NOSNIFF

http://django-secure.rtfd.org/latest/settings.html#secure-content-type-nosniff

SECURE_BROWSER_XSS_FILTER = Value: True
Env variable: DJANGO_SECURE_BROWSER_XSS_FILTER

http://django-secure.rtfd.org/latest/settings.html#secure-browser-xss-filter

INSTALLED_APPS
Appends `djangosecure` to list of `INSTALLED_APPS`.

MIDDLEWARE_CLASSES
Appends `djangosecure` to list of `INSTALLED_APPS`.

DjangoSSLSecurity

class `common_configs.django.security.`**DjangoSSLSecurity**
Adds SSL-related settings to `DjangoSecurity`

SESSION_COOKIE_SECURE = Value: True
Env variable: DJANGO_SESSION_COOKIE_SECURE

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-SESSION_COOKIE_SECURE

CSRF_COOKIE_SECURE = Value: True
Env variable: DJANGO_CSRF_COOKIE_SECURE

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-CSRF_COOKIE_SECURE

SECURE_HSTS_SECONDS = Value: 2592000
Env variable: DJANGO_SECURE_HSTS_SECONDS

http://django-secure.rtfd.org/latest/settings.html#secure-hsts-seconds

SECURE_HSTS_INCLUDE_SUBDOMAINS = Value: True
Env variable: DJANGO_SECURE_HSTS_INCLUDE_SUBDOMAINS

http://django-secure.rtfd.org/latest/settings.html#secure-hsts-include-subdomains

SECURE_SSL_REDIRECT = Value: True
Env variable: DJANGO_SECURE_SSL_REDIRECT

http://django-secure.rtfd.org/latest/settings.html#secure-ssl-redirect

MozillaCSP

class common_configs.django.security.**MozillaCSP**

Mozilla Content Security Policy that defines several default policies for scripts and static assets.

You most probably will want to adjust it according to static storage and CDN usage to white-list all static files sources.

CSP_DEFAULT_SRC()

By default allows `"'self'", STATIC_URL host`

CSP_IMG_SRC()

By default allows `"*", "data:"`

CSP_SCRIPT_SRC()

By default allows `"'self'", "https://ajax.googleapis.com", "https://code.jquery.com", "https://netdna.bootstrapcdn.com", "'unsafe-inline'", STATIC_URL host`

CSP_FONT_SRC()

By default allows `"'self'", "https://themes.googleusercontent.com", "https://netdna.bootstrapcdn.com", STATIC_URL host`

CSP_STYLE_SRC()

By default allows: `"'self'", "https://fonts.googleapis.com", "https://netdna.bootstrapcdn.com", "'unsafe-inline'", STATIC_URL host`

CSP_REPORT_ONLY = BooleanValue

MIDDLEWARE_CLASSES

Appends `djangosecure` to list of `INSTALLED_APPS`.

SingleSite

Single site config.

Set site id, domain name, default emails and allowed hosts.

class common_configs.django.site.**SingleSite**

SITE_ID = Value: 1

Env variable: DJANGO_SITE_ID

https://docs.djangoproject.com/en/dev/ref/settings/#site-id

DOMAIN_NAME = SecretValue

Env variable: DJANGO_DOMAIN_NAME

Default domain name (for email settings, allowed hosts list and session cookie domain)

SITE_NAME = SecretValue

Env variable: DJANGO_SITE_NAME

Default site name (for email name settings)

DEFAULT_FROM_EMAIL = Value: self.get_default_from_email()

Env variable: DJANGO_DEFAULT_FROM_EMAIL

Default: `info@<domain name>`

SERVER_EMAIL = Value: self.get_server_email()
Env variable: DJANGO_SERVER_EMAIL

Default: `server@<domain name>`

EMAIL_SUBJECT_PREFIX = Value: self.get_email_subject_prefix()
Env variable: DJANGO_EMAIL_SUBJECT_PREFIX

Default: `[site name]`

ALLOWED_HOSTS = Value: self.get_allowed_hosts()
Env variable: DJANGO_ALLOWED_HOSTS

Default: `<domain name>`, `www.<domain name>`, `api.<domain name>`

SESSION_COOKIE_DOMAIN = Value: self.get_session_cookie_domain()
Env variable: DJANGO_SESSION_COOKIE_DOMAIN

Default: `<domain name>`

Locale

Django locale, languages and translations

class `common_configs.django.locale.`**Locale**

TIME_ZONE = Value: 'UTC'
Env variable: DJANGO_TIME_ZONE

Default timezone

https://docs.djangoproject.com/en/dev/ref/settings/#time-zone

LANGUAGE_CODE = Value: 'en-us'
Env variable: DJANGO_LANGUAGE_CODE

https://docs.djangoproject.com/en/dev/ref/settings/#language-code

USE_I18N = Value: True
Env variable: DJANGO_USE_I18N

https://docs.djangoproject.com/en/dev/ref/settings/#use-i18n

USE_L10N = Value: True
Env variable: DJANGO_USE_L10N

https://docs.djangoproject.com/en/dev/ref/settings/#use-l10n

USE_TZ = True
https://docs.djangoproject.com/en/dev/ref/settings/#use-tz

1.7.3 Common apps setting mixins

Celery

Celery settings for dev and base production environment.

See http://docs.celeryproject.org/en/latest/configuration.html for settings description.

CeleryBase

class common_configs.apps.celery.**CeleryBase**

CELERY_TIMEZONE = 'UTC'
Timezone

CELERYD_CONCURRENCY = Value: 12
Env variable: CELERYD_CONCURRENCY

The number of concurrent worker processes/threads/green threads executing tasks.

CELERYD_PREFETCH_MULTIPLIER = Value: 1
Env variable: CELERYD_PREFETCH_MULTIPLIER

How many messages to prefetch at a time multiplied by the number of concurrent processes.

CELERY_RESULT_BACKEND = Value
Env variable: DJANGO_CELERY_RESULT_BACKEND

The backend used to store task results (tombstones). Disabled by default.

CELERY_RESULT_SERIALIZER = Value: 'json'
Env variable: DJANGO_CELERY_RESULT_SERIALIZER

Result serialization format.

BROKER_URL = Value
Env variable: CELERY_BROKER_URL

Default broker URL.

BROKER_POOL_LIMIT = Value: 3
Env variable: CELERY_BROKER_POOL_LIMIT

The maximum number of connections that can be open in the connection pool.

CELERY_IGNORE_RESULT = Value: True
Env variable: DJANGO_CELERY_IGNORE_RESULT

Whether to store the task return values or not (tombstones).

CELERY_MESSAGE_COMPRESSION = 'gzip'
Default compression used for task messages.

CELERY_TASK_RESULT_EXPIRES = 1800.0
Time (in seconds, or a timedelta object) for when after stored task tombstones will be deleted.

CELERY_ACKS_LATE = True
Late ack means the task messages will be acknowledged after the task has been executed, not just before, which is the default behavior. https://docs.celeryproject.org/en/latest/faq.html#faq-acks-late-vs-retry

CELERYD_MAX_TASKS_PER_CHILD = Value: 200
Env variable: DJANGO_CELERYD_MAX_TASKS_PER_CHILD

Maximum number of tasks a pool worker process can execute before it's replaced with a new one. Default is no limit.

CELERYD_TASK_TIME_LIMIT = Value: 90
Env variable: DJANGO_CELERYD_TASK_TIME_LIMIT

Task hard time limit in seconds. The worker processing the task will be killed and replaced with a new one when this is exceeded.

CELERY_SEND_TASK_ERROR_EMAILS = Value: True
Env variable: DJANGO_CELERY_SEND_TASK_ERROR_EMAILS

Errors occurring during task execution will be sent to ADMINS by email.

CELERYD_HIJACK_ROOT_LOGGER = False

CELERYBEAT_SCHEDULE = {}

CeleryDev

class common_configs.apps.celery.**CeleryDev**
Bases: common_configs.apps.celery.CeleryBase

CELERY_ALWAYS_EAGER = True
Run queued tasks immediately

CELERY_EAGER_PROPAGATES_EXCEPTIONS = True
Propagate exceptions to front-end debugger

Django Compressor

Assets compression settings for django_compressor

CompressBase

Common settings for production and dev config

class common_configs.apps.compress.**CompressBase**

COMPRESS_ENABLED = Value: True
Env variable: DJANGO_COMPRESS_ENABLED

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_ENABLED

COMPRESS_PARSER = Value: 'compressor.parser.default_htmlparser.DefaultHtmlParser'
Env variable: DJANGO_COMPRESS_PARSER

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_PARSER

COMPRESS_PRECOMPILERS = [('text/coffeescript', 'coffee –compile –stdio'), ('text/less', 'lessc {infile}'), ('text/x-sass', 'sa
Predefined list of precompilers. Require installation of 3rd party binaries and packages

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_PRECOMPILERS

COMPRESS_CSS_FILTERS = Value: ['compressor.filters.css_default.CssAbsoluteFilter', 'compressor.filters.cssmin.CSSM
Env variable: DJANGO_COMPRESS_CSS_FILTERS

List of css filters (rewrite urls and minify output)

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_CSS_FILTERS

COMPRESS_JS_FILTERS = Value: ['compressor.filters.jsmin.JSMinFilter']
Env variable: DJANGO_COMPRESS_JS_FILTERS

List of js filters (minify output)

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_JS_FILTERS

COMPRESS_OUTPUT_DIR = Value: 'cache'
Env variable: DJANGO_COMPRESS_OUTPUT_DIR

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_OUTPUT_DIR

COMPRESS_CACHE_BACKEND = Value: 'locmem'
Env variable: DJANGO_COMPRESS_CACHE_BACKEND

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_CACHE_BACKEND

INSTALLED_APPS
Appends `compressor` to list of `INSTALLED_APPS`.

CompressProd

class common_configs.apps.compress.**CompressProd**
Bases: common_configs.apps.compress.CompressBase

COMPRESS_OFFLINE = Value: True
Env variable: DJANGO_COMPRESS_OFFLINE

Compress assets during deployment

http://django-compressor.rtfd.org/latest/settings/#django.conf.settings.COMPRESS_OFFLINE

CompressDev

class common_configs.apps.compress.**CompressDev**
Bases: common_configs.apps.compress.CompressBase

COMPRESS_CSS_FILTERS = Value: ['compressor.filters.css_default.CssAbsoluteFilter']
Env variable: DJANGO_COMPRESS_CSS_FILTERS

Don't minify css in dev

COMPRESS_JS_FILTERS = Value
Env variable: DJANGO_COMPRESS_JS_FILTERS

Don't minify concatenated js scripts in dev

COMPRESS_CACHE_BACKEND = Value: 'dummy'
Env variable: DJANGO_COMPRESS_CACHE_BACKEND

Don't cache compiled assets during development

Django Debug Toolbar

DebugToolbar

class common_configs.apps.debugtoolbar.**DebugToolbar**

MIDDLEWARE_CLASSES
Appends `debug_toolbar.middleware.DebugToolbarMiddleware` to list of `MIDDLEWARE_CLASSES`.

INSTALLED_APPS
Appends `debug_toolbar` to list of `INSTALLED_APPS`.

Auth

DjangoAuth

class common_configs.apps.auth.**DjangoAuth**

INSTALLED_APPS
Appends `django.contrib.auth` to list of `INSTALLED_APPS`.

MIDDLEWARE_CLASSES
Appends `django.contrib.sessions.middleware.SessionMiddleware` and `django.contrib.auth.middleware.AuthenticationMiddleware` to list of `MIDDLEWARE_CLASSES`.

TEMPLATE_CONTEXT_PROCESSORS
Appends auth context processors to list of `TEMPLATE_CONTEXT_PROCESSORS`.

PASSWORD_HASHERS = ['django.contrib.auth.hashers.BCryptPasswordHasher', 'django.contrib.auth.hashers.PBKDF2P

AllAuth

class common_configs.apps.auth.**AllAuth**
Bases: `common_configs.apps.auth.DjangoAuth`

INSTALLED_APPS
Appends `allauth`, `allauth.account` and `allauth.socialaccount` to list of `INSTALLED_APPS`.

TEMPLATE_CONTEXT_PROCESSORS
Appends allauth context processors to list of `TEMPLATE_CONTEXT_PROCESSORS`.

ACCOUNT_AUTHENTICATION_METHOD = 'email'

ACCOUNT_EMAIL_REQUIRED = True

ACCOUNT_EMAIL_VERIFICATION = 'mandatory'

ACCOUNT_USERNAME_BLACKLIST = ['www', 'ns1', 'ns2', 'ns3', 'ns4', 'ns5', 'dns', 'http', 'https', 'news', 'nntp', 'ftp', 's

AUTHENTICATION_BACKENDS = ['django.contrib.auth.backends.ModelBackend', 'allauth.account.auth_backends.Auth

Django Crispy Forms

Settings for django-crispy-forms - the best way to have DRY Django forms

CrispyForms

class common_configs.apps.forms.**CrispyForms**
Bases: `object`

CRISPY_TEMPLATE_PACK = Value: 'bootstrap3'
Env variable: DJANGO_CRISPY_TEMPLATE_PACK

Template pack

CRISPY_FAIL_SILENTLY = BooleanValue
Env variable: DJANGO_CRISPY_FAIL_SILENTLY

Don't suppress errors unless explicitly set

INSTALLED_APPS
Appends `crispy_forms` to list of `INSTALLED_APPS`.

Django Image Kit

Imagekit

Settings for django-imagekit - automated image processing for Django

class `common_configs.apps.imagekit.`**Imagekit**

IMAGEKIT_DEFAULT_CACHEFILE_STRATEGY = Value: 'imagekit.cachefiles.strategies.Optimistic'
Env variable: IMAGEKIT_DEFAULT_CACHEFILE_STRATEGY

Use optimistic strategy

http://django-imagekit.rtfd.org/latest/configuration.html#django.conf.settings.IMAGEKIT_DEFAULT_CACHEFILE_STRA

IMAGEKIT_SPEC_CACHEFILE_NAMER = Value: 'imagekit.cachefiles.namers.source_name_dot_hash'
Env variable: IMAGEKIT_SPEC_CACHEFILE_NAMER

Define naming strategy

http://django-imagekit.rtfd.org/latest/configuration.html#django.conf.settings.IMAGEKIT_SPEC_CACHEFILE_NAMER

INSTALLED_APPS
Appends `imagekit` to list of `INSTALLED_APPS`.

1.7.4 Common Services mixins

Amazon AWS settings

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

AWS

class `common_configs.services.aws.`**AWS**

AWS_ACCESS_KEY_ID = SecretValue
Env variable: AWS_ACCESS_KEY_ID

AWS access key id retrieved from `AWS_ACCESS_KEY_ID` environment setting

AWS_SECRET_ACCESS_KEY = SecretValue
Env variable: AWS_SECRET_ACCESS_KEY

AWS secret key retrieved from `AWS_SECRET_ACCESS_KEY` environment setting

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>
```

Cache settings

Cache config

CacheDev

class common_configs.services.cache.**CacheDev**
Defines list of caches for development purposes:

- default: DummyCache
- locmem: LocMemCache
- dummy: DummyCache

CACHES = {u'default': {u'BACKEND': u'django.core.cache.backends.dummy.DummyCache'}, u'dummy': {u'BACKENI

Database settings

Database

class common_configs.services.db.**Database**

DATABASES = DatabaseURLValue
Env variable: DATABASE_URL

Configures Django DATABASES using DATABASE_URL environment variable

Email settings

Retrieves email backend config from environment variables

Mailgun

class common_configs.services.mail.**Mailgun**
Use Mailgun as email backend

MAILGUN_API_KEY = SecretValue
Env variable: MAILGUN_API_KEY

Mailgun api key for using the REST API

MAILGUN_SMTP_SERVER = Value: 'smtp.mailgun.org'
Env variable: MAILGUN_SMTP_SERVER

Mailgun SMTP server host

MAILGUN_SMTP_LOGIN = SecretValue
Env variable: MAILGUN_SMTP_LOGIN

Mailgun SMTP server login

MAILGUN_SMTP_PASSWORD = SecretValue
Env variable: MAILGUN_SMTP_PASSWORD

Mailgun SMTP server password

MAILGUN_SMTP_PORT = Value: 587
Env variable: MAILGUN_SMTP_PORT

Mailgun SMTP server port

EMAIL_HOST = Value: 'smtp.mailgun.org'
Env variable: MAILGUN_SMTP_SERVER

Email host for sending mail

EMAIL_HOST_USER = SecretValue
Env variable: MAILGUN_SMTP_LOGIN

Email server username

EMAIL_HOST_PASSWORD = SecretValue
Env variable: MAILGUN_SMTP_PASSWORD

Email server password

EMAIL_PORT = Value: 587
Env variable: MAILGUN_SMTP_PORT

Email server SMTP port

EMAIL_USE_TLS = True
Email server TLS secure connection

EMAIL_BACKEND = Value: 'django.core.mail.backends.smtp.EmailBackend'
Env variable: DJANGO_EMAIL_BACKEND

Email backend

Sendgrid

class common_configs.services.mail.**Sendgrid**
Use Sendgrid as email backend

SENDGRID_SMTP_SERVER = Value: 'smtp.sendgrid.net'
Env variable: SENDGRID_SMTP_SERVER

Sendgrid SMTP server host

SENDGRID_USERNAME = SecretValue
Env variable: SENDGRID_USERNAME

Sendgrid SMTP server login

SENDGRID_PASSWORD = SecretValue
Env variable: SENDGRID_PASSWORD

Sendgrid SMTP server password

SENDGRID_SMTP_PORT = Value: 587
Env variable: SENDGRID_SMTP_PORT

Sendgrid SMTP server port

EMAIL_HOST = Value: 'smtp.sendgrid.net'
Env variable: SENDGRID_SMTP_SERVER

Email host for sending mail

EMAIL_HOST_USER = SecretValue
Env variable: SENDGRID_USERNAME

Email server username

EMAIL_HOST_PASSWORD = SecretValue
Env variable: SENDGRID_PASSWORD

Email server password

EMAIL_PORT = Value: 587
Env variable: SENDGRID_SMTP_PORT

Email server SMTP port

EMAIL_USE_TLS = True
Email server TLS secure connection

EMAIL_BACKEND = Value: 'django.core.mail.backends.smtp.EmailBackend'
Env variable: DJANGO_EMAIL_BACKEND

Email backend

Push notification settings

Configure push notification credentials for Apple Push Notification Service (APNS) Production/Enterprise and Google Cloud Messaging for Android (GCM)

APNS

class common_configs.services.push.**APNS**

APNS_PRODUCTION_SERVER = Value: 'push_production'
Env variable: APNS_PRODUCTION_SERVER

Production server

APNS_PRODUCTION_FEEDBACK_SERVER = Value: 'feedback_production'
Env variable: APNS_PRODUCTION_FEEDBACK_SERVER

Production feedback server

APNS_PRODUCTION_CERT_FILE = Value
Env variable: APNS_PRODUCTION_CERT_FILE

Production certificate file (when cert file is stored in file system)

APNS_PRODUCTION_CERT_STRING = Value
Env variable: APNS_PRODUCTION_CERT_STRING

Production certificate string (when cert is stored as env variable)

APNS_PRODUCTION_CERT_PASS = Value
Env variable: APNS_PRODUCTION_CERT_PASS

Production certificate password

APNS_ENTERPRISE_SERVER = Value: 'push_production'
Env variable: APNS_ENTERPRISE_SERVER

Enterprise server

APNS_ENTERPRISE_FEEDBACK_SERVER = Value: 'feedback_production'
Env variable: APNS_ENTERPRISE_FEEDBACK_SERVER

Enterprise feedback server

APNS_ENTERPRISE_CERT_FILE = Value
Env variable: APNS_ENTERPRISE_CERT_FILE

Enterprise certificate file (when cert file is stored in file system)

APNS_ENTERPRISE_CERT_STRING = Value
Env variable: APNS_ENTERPRISE_CERT_STRING

Enterprise certificate string (when cert is stored as env variable)

APNS_ENTERPRISE_CERT_PASS = Value
Env variable: APNS_ENTERPRISE_CERT_PASS

Enterprise certificate password

APNS_CONFIGS
Dictionary of production and enterprise settings:

```
{
    "production": {
        "server": ..,
        "feedback_server": ..,
        "cert_file": ..,
        "cert_string": ..,
        "cert_pass": ..,
    },
    "enterprise": ..
}
```

Return type dict

GCM

class common_configs.services.push.**GCM**

GCM_API_KEY = SecretValue
Env variable: GCM_API_KEY

API key

GCM_DRY_RUN = BooleanValue
Env variable: GCM_DRY_RUN

Dry run flag

Pusher settings

Pusher service config

Pusher

class common_configs.services.pusherconf.**Pusher**

PUSHER_SOCKET_URL = Value
Env variable: PUSHER_SOCKET_URL

Pusher Socket URL

PUSHER_URL = Value
Env variable: PUSHER_URL

Pusher endpoint URL

PUSHER_APP_ID
Pusher app id from PUSHER_URL

PUSHER_KEY
Pusher key from PUSHER_URL

PUSHER_SECRET
Pusher secret from PUSHER_URL

PUSHER_HOST
Pusher host from PUSHER_URL

Sentry settings

Sentry/Raven configs

Raven

class common_configs.services.sentry.**Raven**

RAVEN_CONFIG = RavenDSNValue
Env variable: SENTRY_DSN

Reads Raven connection settings from SENTRY_DSN environment variable

MIDDLEWARE_CLASSES
Appends raven.contrib.django.raven_compat.middleware.Sentry404CatchMiddleware to list of MIDDLEWARE_CLASSES.

INSTALLED_APPS
Appends raven.contrib.django.raven_compat to list of INSTALLED_APPS.

Twilio settings

Twilio account config

Twilio

class common_configs.services.twilio.**Twilio**

TWILIO_ACCOUNT_SID = SecretValue
Env variable: TWILIO_ACCOUNT_SID

Account SID

TWILIO_AUTH_TOKEN = SecretValue
Env variable: TWILIO_AUTH_TOKEN

Auth token

TWILIO_PHONE_NUMBER = SecretValue
Env variable: TWILIO_PHONE_NUMBER

Default phone number

TWILIO_SKIP_SIGNATURE_VALIDATION = BooleanValue

TWILIO_CALLBACK_DOMAIN = Value

TWILIO_CALLBACK_USE_HTTPS = Value: True

1.7.5 Storage settings and backends

Storage settings

Storage settings for static and media files.

SimpleStorage

class common_configs.storage.storage.**SimpleStorage**
Bases: object

Base class for static and media storage settings

STATIC_URL = '/static/'

MEDIA_URL = '/media/'

BaseCompressStorage

class common_configs.storage.storage.**BaseCompressStorage**
Bases: common_configs.storage.storage.SimpleStorage

Enables django-compress static files finder

STATICFILES_FINDERS = Value: ['django.contrib.staticfiles.finders.FileSystemFinder', 'django.contrib.staticfiles.finde
Env variable: DJANGO_STATICFILES_FINDERS

https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-STATICFILES_FINDERS

AWSCompressStorage

class common_configs.storage.storage.**AWSCompressStorage**
Bases: common_configs.services.aws.AWS, common_configs.storage.storage.BaseCompressStorage

S3 AWS based storage settings configured for use with django_compressor.

For more info see:

- https://docs.djangoproject.com/en/dev/ref/settings/#default-file-storage
- http://django-compressor.readthedocs.org/en/latest/settings/#django.conf.settings.COMPRESS_STORAGE
- https://bitbucket.org/david/django-storages/src/tip/storages/backends/s3boto.py

DEFAULT_FILE_STORAGE = Value: 'common_configs.storage.backends.S3MediaStorage'
Env variable: DJANGO_DEFAULT_FILE_STORAGE

https://docs.djangoproject.com/en/dev/ref/settings/#default-file-storage

STATICFILES_STORAGE = Value: 'common_configs.storage.backends.CachedS3StaticStorage'
Env variable: DJANGO_STATICFILES_STORAGE

https://docs.djangoproject.com/en/dev/ref/settings/#staticfiles-storage

COMPRESS_STORAGE = Value: 'common_configs.storage.backends.CachedS3StaticStorage'
Env variable: DJANGO_COMPRESS_STORAGE

http://django-compressor.readthedocs.org/en/latest/settings/#django.conf.settings.COMPRESS_STORAGE

AWS_STORAGE_BUCKET_NAME = SecretValue
Env variable: AWS_STORAGE_BUCKET_NAME

S3 bucket name for stored files

AWS_DEFAULT_ACL = Value: 'public-read'
Env variable: DJANGO_AWS_DEFAULT_ACL

Default S3 ACL settings

AWS_S3_FILE_OVERWRITE = Value: True
Env variable: AWS_S3_FILE_OVERWRITE

Overwrite files in S3 storage

AWS_S3_SECURE_URLS = Value: True
Env variable: AWS_S3_SECURE_URLS

Use secure SSL urls for serving objects from S3

AWS_S3_CUSTOM_DOMAIN = Value
Env variable: AWS_S3_CUSTOM_DOMAIN

Use custom S3 domain

AWS_CLOUDFRONT_URL = Value
Env variable: AWS_CLOUDFRONT_URL

CloudFront endpoint url when serving files from CDN

AWS_QUERYSTRING_AUTH = BooleanValue
Env variable: AWS_QUERYSTRING_AUTH

Generate S3 auth querystring

AWS_IS_GZIPPED = BooleanValue
Env variable: AWS_IS_GZIPPED

Compress uploaded content

AWS_PRELOAD_METADATA = Value: True
Env variable: AWS_PRELOAD_METADATA

Preload S3 metadata

GZIP_CONTENT_TYPES = Value: ('text/css', 'application/javascript', 'application/x-javascript')
Env variable: AWS_GZIP_CONTENT_TYPES

Compressable content types (if `AWS_IS_GZIPPED` flag is set)

AWS_S3_CALLING_FORMAT
S3 calling format (`SubdomainCallingFormat`).

AWS_HEADERS
Defines far-future expires (2020-12-31) and cache control (`public, max-age=604800, must-revalidate`) headers for served files.

STATIC_URL
Base url for static files

MEDIA_URL
Base url for media files

INSTALLED_APPS
Appends `collectfast` to list of `INSTALLED_APPS`.

LocalCompressStorage

class common_configs.storage.storage.**LocalCompressStorage**
Bases: common_configs.storage.storage.BaseCompressStorage

Storage backends

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

Django config

Remember to configure and include *Amazon AWS settings* config in your settings as well enable user environment variables in the Slug compilation 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.

Available storage backends

S3 Static and Media storage
class common_configs.storage.backends.**S3StaticStorage**(**args*, ***kwargs*)
Subclasses `storages.backends.s3boto.S3BotoStorage` and sets base location for files to `/static`.
class common_configs.storage.backends.**S3MediaStorage**(**args*, ***kwargs*)
Subclasses `storages.backends.s3boto.S3BotoStorage` and sets base location for files to `/media`.

class common_configs.storage.backends.**NonDeletingS3MediaStorage**(**args*, ***kwargs*)

Django Compressor-compatible storage

class common_configs.storage.backends.**CachedS3BotoStorage**(**args*, ***kwargs*)

S3 storage backend that saves the files both remotely and locally.

See http://django_compressor.readthedocs.org/en/latest/remote-storages/

class common_configs.storage.backends.**CachedS3StaticStorage**(**args*, ***kwargs*)

Mix of the S3StaticStorage and CachedS3BotoStorage, saves files in /static subdirectory

1.7.6 Logging patterns

Standard library logging

Logging

Django logging configuration

Defined loggers:

- <catch all>
- django
- django.startup
- django.request
- django.db.backends
- django.commands
- django.security.DisallowedHost
- app.* - for project app loggers
- boto
- celery
- requests
- raven
- sentry.errors

Defined handlers:

- mail_admins
- console
- console_celery
- sentry

class common_configs.logging.stdlib.**Logging**

LOGGING_DEFAULT_HANDLER = Value: 'console'

Env variable: DJANGO_LOGGING_DEFAULT_HANDLER

Default handler

LOGGING_CELERY_HANDLER = Value: 'console_celery'
Env variable: DJANGO_LOGGING_CELERY_HANDLER

Default handler for celery

LOGGING_DEFAULT_FORMATTER = Value: 'console'
Env variable: DJANGO_LOGGING_DEFAULT_FORMATTER

Default formatter

LOGGING_ADD_REQUEST_ID = Value: True
Env variable: DJANGO_LOGGING_ADD_REQUEST_ID

Add request-id to each log line (requires https://github.com/dabapps/django-log-request-id)

LOGGING_USE_SENTRY = Value: True
Env variable: DJANGO_LOGGING_USE_SENTRY

Use sentry for error logging

LOGGING()
Fully configured Django logging

Structlog logging

Enables structlog.org logging.

StructLoggingDev

class common_configs.logging.structlog.**StructLoggingDev**

STRUCTLOG_CONFIGURED

StructLoggingProd

class common_configs.logging.structlog.**StructLoggingProd**

STRUCTLOG_CONFIGURED

1.7.7 PaaS Platform settings

Heroku

PostgresHeroku

class common_configs.paas.heroku.**PostgresHeroku**

DATABASES
Return a fully configured Django DATABASES setting.

Scans environment variables for available postgresql add-on.

```
heroku addons:add heroku-postgresql
heroku addons:add pgbackups:auto-month
```

MemcacheHeroku

class common_configs.paas.heroku.**MemcacheHeroku**

CACHES
Return a fully configured Django CACHES setting.

Scans environment variables for available memcache add-on. Additionally includes Django's LocMemCache backend under "locmem" cache name.

```
heroku addons:add memcachier
```

CeleryHerokuBigWig

Uses AMQP broker settings for RabbitMQ Bigwig add-on.

class common_configs.paas.heroku.**CeleryHerokuBigWig**
Bases: common_configs.apps.celery.CeleryBase

BROKER_URL = SecretValue
Env variable: RABBITMQ_BIGWIG_URL

Retrieve broker url settings from BigWig RabbitMQ environment variable

```
heroku addons:add rabbitmq-bigwig
```

CeleryHerokuCloudAMQP

Uses AMQP broker settings for CloudAMQP add-on.

class common_configs.paas.heroku.**CeleryHerokuCloudAMQP**
Bases: common_configs.apps.celery.CeleryBase

BROKER_URL = SecretValue
Env variable: CLOUDAMQP_URL

Retrieve broker url settings from CloudAMQP environment variable

```
heroku addons:add cloudamqp
```

Heroku

class common_configs.paas.heroku.**Heroku**
Bases: common_configs.paas.heroku.MemcacheHeroku, common_configs.paas.heroku.PostgresHerok
common_configs.storage.storage.AWSCompressStorage

SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
SSL proxy header as defined in Django settings docs.

LOG_REQUEST_ID_HEADER = 'HTTP_HEROKU_REQUEST_ID'
Request Id header

1.7.8 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/nigma/django-common-configs/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

django-common-configs could always use more documentation, whether as part of the official django-common-configs docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/nigma/django-common-configs/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *django-common-configs* for local development.

1. Fork the *django-common-configs* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-common-configs.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-common-configs
$ cd django-common-configs/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8:

```
$ flake8 common_configs
```

To get flake8, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
2. The pull request should work for Python 2.7, and 3.3, and for PyPy.

1.7.9 Credits

Development Lead

- Filip Wasilewski <en@ig.ma>

Contributors

None yet. Why not be the first?

1.7.10 History

0.1.0 (2014-mm-dd)

- First release

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

a

b

c

d

h

l

m

p

s

t