Combinable Settings

Aymeric Augustin - @aymericaugustin

DjangoCon US - September 5th, 2012

don’t write large monolithic apps

well

it’s too late

Problem

- Multiple sites
- Multiple environments
- Different kinds settings
- Lots of settings!

Problem

How to obtain (easily)
the settings for the www site
in the dev environment?

or any other permutation?

Solution

```
$ django-admin.py runserver
    --settings=settings.presets.dev_www
```

Files

```
settings
├── __init__.py
├── available
│   ├── base.py
│   ├── dev.py
│   ├── prod.py
│   ├── ws.py
│   └── www.py
├── presets
│   ├── dev_ws.py
│   ├── dev_www.py
│   ├── prod_ws.py
│   └── prod_www.py
└── util.py
```

The code

```
# settings/presets/dev_www.py

from settings.util import load_settings

load_settings(globals(),
              'base', 'dev', 'www')
```

The code

```
# settings/available/base.py

INSTALLED_APPS = (
    ...,
)

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

The code

```
# settings/available/www.py

INSTALLED_APPS += (
    ...,
)

SITE_ID = 1

ROOT_URLCONF = 'www.urls'
```

The code

```
# settings/available/dev.py

CACHES = {
    'default': {...},
}

DATABASES = {
    'default': {...},
}

SECRET_KEY = 'whatever, only for dev'
```

Default settings

```
# settings/util.py

from os.path import abspath, dirname, join

ROOT_DIR = abspath(dirname(dirname(__file__)))

def load_settings(env, *settings):
    for s in settings:
        f = join(ROOT_DIR, 'settings',
                 'available', '%s.py' % s)
        execfile(f, env)
```

Default settings

```
# settings/__init__.py

import os
from django.conf import ENVIRONMENT_VARIABLE

if os.environ.get(ENVIRONMENT_VARIABLE) \
        == __name__:
    from settings.util import load_enabled
    load_enabled(globals())
```

Default settings

```
# settings/util.py

from glob import iglob
from os.path import basename, join, realpath
```

```
def load_enabled(env):
    files = iglob(join(ROOT_DIR, 'settings',
                       'enabled', '*.py'))
    files = sorted(basename(realpath(f))[:-3]
                   for f in files)
    load_settings(env, *files)
```

Local settings

```
# settings/util.py

def load_settings(env, *settings):
    for s in settings:
        f = join(ROOT_DIR, 'settings',
                 'available', '%s.py' % s)
        execfile(f, env)

    for f in iglob(join(ROOT_DIR, 'settings',
                        'local', '*.py')):
        execfile(f, env)
```

Advantages

- Flexible
- Scalable
  - dozens of presets
  - hundreds of settings

Drawback

- Incremental combination requires using execfile rather than import
- Autoreload doesn’t pick changes
- Global variables are implicitly passed along the chain of files
- Code becomes ugly on Python 3

Thank you

please use responsibly