

Déploiement d'une application Django

Author : Frédéric de Zorzi

Contact : fredz@nospam.pimentech.net

Revision : \$Revision : 8419 \$

Date : \$Date : 2010-02-26 16 :26 :59 +0100 (Fri, 26 Feb 2010) \$

Copyright : This document has been placed in the public domain.

Tags : django system apache

OUTDATED

Il y a trois choses à dissocier dans un projet Django :

1. Les applications
2. La conf du site : le fichier `setting.py`
3. Les templates propres au site

Du point de vue Apache

```
<Location "/">
    # On utilise mod_python pour ce chemin :
    SetHandler python-program

    # Qui lui même utilise Django :
    PythonHandler django.core.handlers.modpython

    # Qui utilise les propriétés de "monsite" :
    SetEnv DJANGO_SETTINGS_MODULE monsite.settings
</location>
```

Emplacement des applications

Les applications pouvant être utilisées par plusieurs sites ou par des scripts python, le meilleur emplacement pour elles est dans `/usr/lib/pythonX.X/site-packages`. Pour nous, le mieux est de les mettre dans `django_pimentech`.

Pour l'installation, on utilise `distutils` :

```
from distutils.core import setup
from distutils.command.install_data import install_data
from distutils.command.install import INSTALL_SCHEMES
import os
import sys

class osx_install_data(install_data):
    # On MacOS, the platform-specific lib dir is /System/Library/Framework/Python,
    # which is wrong. Python 2.5 supplied with MacOS 10.5 has an Apple-specific fix
    # for this in distutils.command.install_data#306. It fixes install_lib but not
    # install_data, which is why we roll our own install_data class.

    def finalize_options(self):
        # By the time finalize_options is called, install.install_lib is set to the
        # fixed directory, so we set the installdir to install_lib. The
        # install_data class uses ('install_data', 'install_dir') instead.
        self.set_undefined_options('install', ('install_lib', 'install_dir'))
        install_data.finalize_options(self)
```

```

if sys.platform == "darwin":
    cmdclasses = {'install_data': osx_install_data}
else:
    cmdclasses = {'install_data': install_data}

def fullsplit(path, result=None):
    """
    Split a pathname into components (the opposite of os.path.join) in a
    platform-neutral way.
    """
    if result is None:
        result = []
    head, tail = os.path.split(path)
    if head == '':
        return [tail] + result
    if head == path:
        return result
    return fullsplit(head, [tail] + result)

# Tell distutils to put the data_files in platform-specific installation
# locations. See here for an explanation:
# http://groups.google.com/group/comp.lang.python/browse_thread/thread/35ec7b2fed
for scheme in INSTALL_SCHEMES.values():
    scheme['data'] = scheme['purelib']

# Compile the list of packages available, because distutils doesn't have
# an easy way to do this.
packages, data_files = [], []
root_dir = os.path.dirname(__file__)
len_root_dir = len(root_dir)
package_dir = os.path.join(root_dir, 'django_pimentech')

for dirpath, dirnames, filenames in os.walk(package_dir):
    # Ignore dirnames that start with '.'
    for i, dirname in enumerate(dirnames):
        if dirname.startswith('.') or dirname.startswith('CVS'): del dirnames[i]
    if '__init__.py' in filenames:
        package = dirpath[len_root_dir:].lstrip('/').replace('/', '.')
        packages.append(package)
    else:
        data_files.append([dirpath, [os.path.join(dirpath, f) for f in filenames]])

# Small hack for working with bdist_wininst.
# See http://mail.python.org/pipermail/distutils-sig/2004-August/004134.html
if sys.argv[1] == 'bdist_wininst':
    for file_info in data_files:
        file_info[0] = '/PURELIB/%s' % file_info[0]

setup(
    name = "libCommonDjango",
    url = 'http://www.pimentech.fr',
    author = 'PimenTech',
    author_email = 'info@pimentech.net',
    description = 'Django libraries from PimenTech.',
    packages = packages,
    data_files = data_files,

```

)

Les templates

Les templates non spécifiques propres aux applications sont dans les sous-répertoires des applications. Pour ce module (libcommonDjango), on n'a rien à faire de spécifique.

Le répertoire des templates est défini dans la variable `TEMPLATE_DIRS`¹. Ces templates doivent avoir un droit de lecture `www-data`, mais ne sont pas lues directement par apache (on n'a pas à les placer dans `/var/www` ou autre). Pour le site "garage" j'ai opté pour :

- `~/templates_django` pour les templates
- `~/lib/python/site_garage` pour la configuration python (principalement `settings.py` et `urls.py`).

¹Cette variable est en fait une liste, on peut définir plusieurs répertoires.