django CMS Installer Documentation

Release 1.0.0.dev2

Iacopo Spalletti

Jul 15, 2017

Contents

Contents:

CHAPTER 1

django CMS Installer

Command to easily bootstrap django CMS projects

- Free software: BSD license

Features

`djangocms-installer` is a console wizard to help bootstrapping a django CMS project.

Refer to django CMS Tutorial on how to properly setup your first django CMS project.

Warning: Version 0.9 dropped support for Python 2.6, Django <1.8 and django CMS < 3.2. More 0.8.x versions may be released after 0.9 is out in case important bugfixes will be needed.

Usage

To create your first django CMS project run:

```
djangocms my_project
```

That's all!

This command will:

- Create a Django project
- Install django CMS and its core plugins
- Create and populate the database
- Install default templates

Just run `manage.py runserver`, go to http://localhost:8000 , login with user *admin* (same password) and enjoy your first django CMS project.

More at django CMS Tutorial and installer usage page

Documentation

For detailed information see https://djangocms-installer.readthedocs.io

Preliminary checks and system libraries

While this wizard try to handle most of the things for you, it doesn't check for all the proper native (non python) libraries to be installed. Before running this, please check you have the proper header and libraries installed and available for packages to be installed.

Libraries you would want to check:

- libjpeg (for JPEG support in `Pillow`)
- zlib (for PNG support in `Pillow`)
- postgresql (for `psycopg2`)
- libmysqlclient (for `Mysql`)
- python-dev (for compilation and linking)

For additional information, check https://djangocms-installer.readthedocs.io/en/latest/libraries.html

Supported versions

The current supported version matrix is the following:

	Django 1.8	Django 1.9	Django 1.10
django CMS 3.2	Supported	Supported	Unsupported
django CMS 3.3	Supported	Supported	Unsupported
django CMS 3.4	Supported	Supported	Supported

See version 0.8 for older Django / django CMS versions support

Any beta and develop version of Django and django CMS, by its very nature, it's not supported, while it still may work.

`djangocms-installer` tries to support beta versions of django CMS when they are be considered sufficiently stable by the upstream project.

Warning

`djangocms-installer` assumes that `django-admin.py` is installed in the same directory as python executable, which is the standard virtualenv layout. Other installation layouts might work, but are not officially supported.

Windows support

The installer is tested on Windows 7 with Python versions 3.4.2 and 2.7.8 installed using official MSI packages available at http://python.org.

Please check that the `.py` extension is associated correctly with Python interpreter:

```
c:\> assoc .py
.py=Python.File

c:\>ftype Python.File
Python.File="C:\Windows\py.exe" "%1" %*
```

CHAPTER 2

Usage

djangocms installer creates a complete and fully featured django CMS project.

By default it:

- creates the project
- installs requirements
- creates database
- (optionally) creates a sample database
- (optionally) copy a provided set of templates
- write the requirements file to the project directory

djangocms installer works as a batch script and as a command line wizard.

Batch mode (default)

In batch mode **djangocms installer** will use the arguments provided to create and configure the project. See the complete list of *Arguments reference* for reference.

```
djangocms my_project
```

Wizard mode

Wizard mode works by asking relevant questions to the user; it can be invoked with `-w` option:

```
djangocms -w -p /path/whatever project_name
```

A wizard will ask for the missing parameters; for most of them sane defaults are provided, but you're free to adapt to your own needs. The only required parameters are the database name, in url format, and the project languages, as a comma separated list.

Dump mode

By using the *-R* arguments, **djangocms-installer** won't create a new django CMS instance but will print to stdout the list of packages required to properly setup the virtualenv. This can be helpful to customize the virtualenv:

1. Dump the list of requirements:

```
$ djangocms -p /path/whatever project_name -R > requirements.txt
```

2. Edit requirements.txt according to your needs
3. Run the installer again providing the customized requirements file:

```
$ djangocms -r custom_requirements.txt -p /path/whatever project_name
```

   or install the requirements manually and execute the installer with *n* argument:

```
$ pip install -r custom_requirements.txt
$ djangocms -n -p /path/whatever project_name
```

See *Arguments reference* for arguments reference

Config file mode

In config file mode, all (or some) options can be provided via an external configuration file.

See a complete example with all available arguments.

Is it possible to either provide all the values in the config file:

```
djangocms --config-file /path/to/config.ini project_name
```

Or just some, or overriding by using the command line arguments:

```
djangocms --config-file /path/general-config.ini -p /path/other/proj -s -q project_
↪name
```

Note: If config.ini not contains *no-input = true* and *-q* argument isn't set then one act as a placeholder with default values for wizard.

Dump config files

Values passed to the installer can be dumped for later reuse:

```
djangocms --config-dump /path/config.ini -p . project_name
```

if installation fails dump can be used to fix some arguments and re-run installer with dumped config:

```
djangocms --config-dump /path/config.ini --db postgres://wrong-usr:pwd@host/db -p .␣
↪project_name
# fails
```

```
djangocms --config-file /path/config.ini --db postgres://correct-user:pwd@host/db -p .
↪ project_name
# succeed
```

Custom settings

If want or need to provide custom settings **before** the initial database sync is run, use *–extra-settings* parameter. To use this option, pass the path to a file as argument: its content is going to be appended to the generated settings file.

HOWTO

1. Create an empty virtualenv:

   ```
   virtualenv /virtualenv/path/my_project
   ```

2. Install *djangocms-installer*:

   ```
   pip install djangocms-installer
   ```

   or:

   ```
   pip install https://github.com/nephila/djangocms-installer/archive/master.zip
   ```

3. Execute the wizard:

   ```
   djangocms project_name
   ```

4. Answer the wizard questions;
5. Change to your project directory:

   ```
   cd /path/whatever project_name
   ```

6. Modify the provided settings. You will want to modify at least the language and the template settings;
7. Execute the project:

   ```
   (whatever) $ python manage.py runserver
   ```

8. Enjoy!

Use different templates directory

You can create the base project with a custom templateset by using the `--templates` parameter. Be aware that while **djangocms installer** will copy the files for you, it won't update the `CMS_TEMPLATES` settings parameter, so you'll need to modify that after installation.

Bare install

You can optionally install just Django and django CMS without any additiona plugin by using the `--no-plugins` option; this will allow you to further customise your installation.

CHAPTER 3

FAQ

1. I need to use version **foo** of package *blargh*, while the installer want to use version **baz**, how can I solve this?

   Use *Dump mode* to dump the requirements used by the installer, customize them and pass them again to the installer for the installation run

2. Packages are not correctly installed / missing in my environment

   If you're using Debina / Ubuntu, chances are that the version of pip in the virtualenv is outdated and cannot connect with PyPi. Usually upgrading pip in the virtualenv before running **djangocms-installer** fixes the issue:

   ```
   pip install -U pip
   ```

3. After installing the virtualenv, the installer exit with "*Pillow is not compiled with ... support*" / "*Pillow is not installed*" errors, what can I do?

   Pillow can be a bit tricky in some environments, so please check the "*Libraries installation issues*" section for more detailed help.

4. I followed the advices above, but I keep having the same messages!

   Checking for correct Pillow installation can be a bit tricky: installer try to be smart but it may sometimes fail and report Pillow errors while errors lie somewhere else. In this case, please open an issue on github project, or ask in the #django-cms IRC channel.

5. How can I configure the database to use?

   djangocms-installer uses dj-database-url to get database configuration parameters; refer to this package for more details.

6. The installer dies with an error like `ImportError: Could not import settings 'foo.bar.settings' (Is it on sys.path? Is there an import error in the settings file?): No module named foo.bar.settings`, what's happening?

   Chances are you have `DJANGO_SETTINGS_MODULE` set in you environment, either by default or using postactivate virtualenv hooks or other tools; please check you environment right after the error happening (for example using the `env` command on *nix systems) and remove any customisation: the installer requires that `DJANGO_SETTINGS_MODULE` is not set on the first run. You can customise it later.

CHAPTER 4

Arguments reference

Required arguments

You must always provide the following argument when invoking **djangocms installer**:

- `project_name`: Name of the project to be created

Optionally you can provide a project directory, otherwise a directory named after the project name will be created in the current directory.

Warning: project directory dir is the main project directory (the one where `manage.py` will be created); by default the installer check if it's empty (minus hidden files) to ensure that you're running in a clean environment. If you want to use a non-empty directory use the `-s` flag;

Wizard arguments

The following arguments can be overridden in *Wizard mode*

- `--db`, `-d`: Database configuration (in URL format); use dj-database-url syntax; default: `sqlite://localhost/project.db`
- `--i18n`, `-i`: Activate Django I18N / L10N setting; choices: `yes|no`, default: `yes`
- `--use-tz`, `-z`: Activate Django timezone support; choices: `yes|no`, default: `yes`
- `--timezone`, `-t`: Optional default time zone, default: *local timezone*
- `--reversion`, `-e`: Install and configure reversion support, choices: `yes|no`, default: `yes`; this will ignored for RC/develop/3.4 versions of django CMS as support has been dropped upstream;
- `--permissions`: Activate CMS permission management; choices: `yes|no`, default: `no`

- `--languages`, `-l`: Languages available for the project. Option can be provided multiple times, or as a comma separated list. Only language codes supported by Django can be used here; refer to django source for a list of supported codes.
- `--django-version`: Django version; choices: `1.8|1.9|1.10|stable|lts`, default: `lts`
- `--cms-version`, `-v`: django CMS version, choices: `3.2|3.3|3.4|stable|lts`. default: `lts`
- `--bootstrap`: Use Twitter Bootstrap as theme, choices: `yes|no`, default: `no`
- `--starting-page`: Load a starting page with examples (available for english language only) after installation, choices: `yes|no`, default: `no`
- `--templates`: Use a custom directory as template source; is checked to be a valid path, otherwise the shipped templates are used

Note: Django `stable` keyword is expanded to latest released Django version

Note: Django `lts` keyword is expanded to latest released Django LTSrsion

Note: django-cms `stable` keyword is expanded to latest released django-cms version

Note: django-cms `lts` keyword is expanded to latest released django-cms version

Warning: if an unsupported combination of Django and django CMS version is selected, the wizard exits reporting the error.

Advanced options

The following options are not managed by the config wizard and are meant for advanced usage:

- `--no-input`, `-q`: If given **djangocms installer** run in *Batch mode (default)* (default behavior);
- `--wizard`, `-w`: If given **djangocms installer** run in *Wizard mode*;
- `--parent-dir`, `-p`: Optional project directory;
- `--verbose`, : Provides output of the commands used to setup the project, namely `pip` and `django-admin`;
- `--filer`, `-f`: Install and configure django-filer plugins; since 0.9 this is enabled by default and can't be disabled;
- `--config-file`: Provides the configuration options via a ini file; see *Config file mode*;
- `--config-dump`: Dumps the configuration in a format suitable for `-config-file` option; see *Config file mode*;
- `--dump-requirements`, `-R`: Dumps the generated requirements to stdout and exits; see *Dump mode*;
- `--requirements`, `-r`: You can use a custom requirements files instead of the requirements provided by **djangocms installer**;

- `--no-deps`, `-n`: Don't install package dependencies;
- `--no-plugins`: Don't install plugins;
- `--no-db-driver`: Don't install database package;
- `--no-sync`, `-m`: Don't run syncdb / migrate after bootstrapping the project;
- `--no-user`, `-u`: Don't create the admin user;
- `--utc`, : Use UTC as default timezone;
- `--list-plugins`, `-P`: List plugins that's going to be installed and configured for the project; this will not alter the virtualenv or create the project;
- `--extra-settings`: Path to a file with extra variables to append to generated settings file. It doesn't need to be a Python file, its content is blindly copied in the project settings.
- `--skip-empty-check`, `-s`: Skip the check if the project dir contains files or directory; in case of error when setting up the project the existing directory will be preserved.
- `--delete-project-dir`‘, `-c`: Delete project directory on creation failure in *Batch mode (default)*.

CHAPTER 5

Libraries installation issues

While this wizard tries to handle most of the things for you, it doesn't check for all the proper native (non python) libraries to be installed. Before running this, please check you have the proper header and libraries installed and available for packages to be installed.

Libraries you would want to check:

- libjpeg (for JPEG support in `Pillow`)
- zlib (for PNG support in `Pillow`)
- postgresql (for `psycopg`)
- libmysqlclient (for `Mysql-Python`)

The actual package name may vary depending on the platform / distribution you are using; you should make sure you have the library headers file installed (mostly contained in package with *-dev* in its name: e.g. *libjpeg-dev* for *libjpeg* library).

Examples

The following are examples based on major distributions versions at the time of writing.

This may may vary over time with distribution changes and may be out of date / unsuitable for you distribution.

The rationale should stand over the time, though, even if package manager and packages may change name and if a package name does not match when your these commands, a quick search using your distribution package manager should help you to find the right name.

Debian / Ubuntu

- Pillow: `sudo apt-get install libtiff5-dev libjpeg8-dev zlib1g-dev libfreetype6-dev`
- Postgres: `sudo apt-get install libpq-dev`

- MySQL: `sudo apt-get install libmysqlclient-dev`

Fedora

- Python: `dnf install python3-devel gcc` (or `dnf install python-devel gcc`)
- Pillow: `dnf install libtiff-devel libjpeg-devel libzip-devel freetype-devel lcms2-devel libwebp-devel zlib-devel`
- Postgres: `dnf install postgresql-devel`
- MySQL: `dnf install mysql-devel` (or `yum install mariadb-devel`)

CentOS / yum based distributions

- Python: `yum install python3-devel gcc` (or `yum install python-devel gcc`)
- Pillow: `yum install libjpeg-devel libpng-devel libtiff-devel freetype-devel zlib-devel`
- Postgres: `yum install postgresql-devel`
- MySQL: `yum install mysql-devel` (or `yum install mariadb-devel`)

Fixing libraries installation issues

If a native library is missing when installing a python package, the package installation may fail silently or the package may be missing some functionality (e.g.: if libjpeg is not installed Pillow will be compiled without JPEG support).

`djangocms-installer` tries to check for most common issues and will exit with an exception in case of errors.

In case of package installation failure you can simply install the correct library and execute `djangocms-installer` again with the same parameters; if the package is compiled with some functionality missing, you have to first deinstall it (`pip uninstall *package-name*`), then install the correct library and the execute `djangocms-installer` again.

- Reinstall `pip install --upgrade --force-reinstall --no-deps djangocms-installer`

For older Debian / Ubuntu releases, a manual fix may be needed to properly fix compilation issues: see stackoverflow

Manual installation

If all else fails, you can use *Dump mode* to create a requirements file, install it by hand and run the installer with `-n` parameter.

CHAPTER 6

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/nephila/djangocms-installer/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 CMS Installer could always use more documentation, whether as part of the official django CMS Installer 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/nephila/djangocms-installer/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 *djangocms-installer* for local development.

1. Fork the *djangocms-installer* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/djangocms-installer.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 djangocms-installer
$ cd djangocms-installer/
$ python setup.py develop
$ pip install -r requirements_dev.txt
```

the last one is to get the requirements including testing and development tools installed.

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 and the tests, including testing other Python versions with tox:

```
$ flake8 djangocms-installer tests
$ python setup.py test
$ tox
```

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. The pull request should include tests.
2. 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.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/nephila/djangocms-installer/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python setup.py test -s tests.main
```

CHAPTER 7

Credits

Development Lead

- Iacopo Spalletti <i.spalletti@nephila.it>
- Kim Thoenen <kim@smuzey.ch>
- Patrick Lauber <digi@treepy.com>

Contributors

- Aaron Boman
- Alexander Pervakov
- Boussard
- Carlo Ascani <c.ascani@nephila.it>
- Claudio Luck
- Comedy
- Enkel Mitrushi
- growlf
- Henning Sprang <henning.sprang@gmail.com>
- Jonas Obrist <ojiidotch@gmail.com>
- Nick Moore <nick@zoic.org>
- pipsqueaker

CHAPTER 8

History

1.0.0 (unreleased)

- Dropped cmsplugin-filer in favor of core plugins which now use filer

0.9.7 (2017-07-15)

- Improve django-admin invocation to support more python environments layouts

0.9.6 (2017-03-12)

- Added more Django 1.10 fixes / cleanups
- Added python 3.6 compatibility

0.9.5 (2017-02-16)

- Added more Django 1.10 fixes / cleanups

0.9.4 (2017-01-03)

- Added support for Django 1.10
- Added 'lts' keyword (it's now the default instead of 'stable')

0.9.3 (2016-11-16)

- Fixed issue with create_user command

0.9.2 (2016-11-12)

- Fixed search 'django-admin.py'
- Improved error reporting during package installation

0.9.1 (2016-10-02)

- Fixed issue with -p parameter

0.9.0 (2016-09-15)

- Drop support for Python 2.6
- Drop support for Django < 1.8
- Drop support for django CMS < 3.2
- Improve inline documentation
- If -s option is used, original directory is not removed
- Set django CMS 3.3 as stable
- Add support for 'rc' releases
- Only "core" plugins are now supported
- Drop support for django-reversion in django CMS 3.4 (due to upstream drop)
- Make project-path option optional
- Batch mode is now the default one
- Add support for conda package manager
- Admin user is now created in batch mode by default. Use –no-user option to skip user creation
- In batch mode directory is not removed in case of cleanup unless –delete-project-dir is given
- Disable permission by default

0.8.12 (2016-08-27)

- Pin plugins versions

0.8.11 (2016-07-15)

- Better plugins version pinning
- Move sitemaps to non-language prefix url
- Fallback to UTC when timezone cannot be detected
- Pin html5lib version

0.8.10 (2016-05-28)

- Add support for django CMS 3.3 final

0.8.9 (2016-05-19)

- Add support for django CMS 3.3rc

0.8.8 (2016-05-06)

- Force language codes to lowercase
- Force i18n if multiple languages is provided
- Fix some errors in selecting dependencies
- Fix error in Django 1.9 regexp

0.8.7 (2016-02-23)

- Add clearer cleanup message

0.8.6 (2016-02-05)

- Add support for Django 1.9
- Fix formatting CONN_MAX_AGE
- Improve error handling in case of fatal error

0.8.5 (2015-12-24)

- Fix createsuperuser command

0.8.4 (2015-12-21)

- Remove flash plugin from installed plugins
- Add `--verbose` option to improve debug
- Fix unicode errors
- Improve documentation

0.8.3 (2015-11-25)

- Improve text editor plugin version selection
- Improve admin style version selection

0.8.2 (2015-11-24)

- Add support for django CMS 3.2
- Add support for apphook reload middleware
- Add viewport meta tag for mobile devices support

0.8.1 (2015-10-11)

- Add option to not install plugins
- Add Python 3.5 to build matrix
- Add argument to pass options to pip
- Fix support for custom user models
- Dump requirements file in the created project
- Improve documentation

0.8.0 (2015-08-30)

- Options can now be provided via an ini file for easy scripting
- Better migration modules discovery strategy
- Minor fixes

0.7.9 (2015-07-21)

- Better Django 1.8 support
- Fix error with newer Pillow versions

0.7.8 (2015-06-27)

- Add Django 1.8 support
- Fix template styles

0.7.7 (2015-06-05)

- Switch to cloudflare CDN for bootstrap template
- Fix support for django-filer 0.9.10

0.7.6 (2015-05-01)

- Switch to django CMS 3.1 as stable django CMS release
- Rework the Django supported matrix
- Always use djangocms-link instead of cmsplugin-filer-link

0.7.5 (2015-04-21)

- Add support for django CMS 3.1
- Switch to Django 1.7 as stable django release

0.7.4 (2015-04-14)

- Add automatic timezone detection
- Pin django-reversion versions
- Make installer more compatible with windows environment

0.7.3 (2015-04-08)

- Fix issues with django CMS requirements
- Fix minor issues in shipped templates

0.7.2 (2015-02-08)

- Fixed Windows compatibility issues
- Fixed python 3 compatibility issues
- Add a flag to skip the project directory emptiness check

0.7.1 (2015-01-15)

- Ask for permission before cleanup
- Clarify the *-p* parameter
- Check if the project directory is empty before proceeding

0.7.0 (2015-01-10)

- Improved support for Django 1.7 and django CMS develop (3.1)
- Totally new test strategy
- Reverted -I parameter to install packages
- Improved support for cleanup after failure

0.6.0 (2014-11-30)

- Add support for installing aldryn-boilerplate
- Force installing all packages (-I) when creating the project virtualenv
- Fix multiplatform support bugs
- Update supported Django / django CMS versions
- Add preliminary support for django CMS develop (3.1)

0.5.4 (2014-08-14)

- Fix reversion version selection for older Django versions
- Better project name validation

0.5.3 (2014-07-23)

- Add support for easy_thumbnails 2.0 migrations
- Fix asking for creating user even when –no-input flag is used
- Unpin reversion as django CMS 3.0.3 solves the issue
- Versioned dependency for django-filer when installing django CMS 2.4
- Switch to official django-filer and cmsplugin-filer releases for CMS 3.0

0.5.2 (2014-05-30)

- Pin reversion to 1.8 waiting for a proper fix in django CMS

0.5.1 (2014-05-22)

- Fix error in bootstrap template handling
- Add clarification about custom template set and starting page

0.5.0 (2014-05-21)

- Add dump-requirements argument
- Add user provided extra setting
- Add FAQ section
- Add templates argument
- Documentation update

0.4.2 (2014-04-26)

- Use current cms.context_processors.cms_settings instead of deprecated one
- Document some fixes for library issues
- Fix Python 3 issue
- Switch default Django version to stable instead of 1.5

0.4.1 (2014-04-09)

- Fix some newlines issues in the settings file

0.4.0 (2014-04-09)

- Update for django CMS 3.0 stable!
- Fixes for settings parameter

0.3.5 (2014-04-03)

- Update for django CMS 3.0c2

0.3.4 (2014-03-29)

- Fix issues with django CMS 2.4

0.3.3 (2014-03-20)

- Better handling of differenct CMS version configuration

0.3.2 (2014-03-18)

- Fix some versioned dependency resolve error

0.3.1 (2014-03-16)

- Fix error in loading resource files
- Fix error with non-standard python executable paths
- Fix error with Django 1.6
- Fix error installing django-filer

0.3.0 (2014-03-15)

- Sync with django CMS RC1 changes
- Use external django CMS plugins instead of removed core ones

0.2.0 (2014-02-06)

- Project renamed to djangocms-installer
- Bugfixes
- Better default templates
- Python 3 compatibility
- Django 1.6 compatibility
- django CMS 3 beta3 and dev snapshot support
- Support for django-admin project templates
- Ships Twitter bootstrap templates
- Can now creates a dummy starting page

0.1.1 (2013-10-20)

- Improved documentation on how to fix installation in case of missing libraries.

0.1.0 (2013-10-19)

- First public release.

CHAPTER 9

Package documentation

djangocms_installer package

Subpackages

djangocms_installer.config package

Submodules

djangocms_installer.config.internal module

class djangocms_installer.config.internal.**DbAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

djangocms_installer.config.internal.**validate_project**(*project_name*)

Check the defined project name against keywords, builtins and existing modules to avoid name clashing

djangocms_installer.config.settings module

Module contents

djangocms_installer.config.**get_settings**()

djangocms_installer.config.**parse**(*args*)

Define the available arguments

djangocms_installer.config.**show_plugins**()

Shows a descriptive text about supported plugins

djangocms_installer.config.**show_requirements**(*args*)
Prints the list of requirements according to the arguments provided

djangocms_installer.config.**write_default**(*config*)

djangocms_installer.django package

Module contents

djangocms_installer.django.**copy_files**(*config_data*)
It's a little rude actually: it just overwrites the django-generated urls.py with a custom version and put other files in the project directory.

Parameters **config_data** – configuration data

djangocms_installer.django.**create_project**(*config_data*)
Call django-admin to create the project structure

Parameters **config_data** – configuration data

djangocms_installer.django.**create_user**(*config_data*)
Create admin user without user input

Parameters **config_data** – configuration data

djangocms_installer.django.**load_starting_page**(*config_data*)
Load starting page into the CMS

Parameters **config_data** – configuration data

djangocms_installer.django.**patch_settings**(*config_data*)
Modify the settings file created by Django injecting the django CMS configuration

Parameters **config_data** – configuration data

djangocms_installer.django.**setup_database**(*config_data*)
Run the migrate command to create the database schema

Parameters **config_data** – configuration data

djangocms_installer.install package

Module contents

djangocms_installer.install.**check_install**(*config_data*)
Here we do some **really** basic environment sanity checks.

Basically we test for the more delicate and failing-prone dependencies:

- database driver
- Pillow image format support

Many other errors will go undetected

djangocms_installer.install.**cleanup**(*requirements*)

djangocms_installer.install.**cleanup_directory**(*config_data*)
Asks user for removal of project directory and eventually removes it

djangocms_installer.install.**requirements**(*requirements*, *pip_options=u''*, *is_file=False*, *verbose=False*)

djangocms_installer.install.**write_requirements**(*config_data*)

Submodules

djangocms_installer.compat module

djangocms_installer.compat.**clean**(*value*)

djangocms_installer.main module

djangocms_installer.main.**execute**()

djangocms_installer.utils module

class djangocms_installer.utils.**chdir**(*newPath*)
Bases: object

Context manager for changing the current working directory

djangocms_installer.utils.**format_val**(*val*)
Returns val as integer or as escaped string according to its value :param val: any value :return: formatted string

djangocms_installer.utils.**less_than_version**(*value*)
Converts the current version to the next one for inserting into requirements in the ‘ < version’ format

djangocms_installer.utils.**query_yes_no**(*question*, *default=None*)
Ask a yes/no question via *raw_input()* and return their answer.

Parameters

- **question** – A string that is presented to the user.
- **default** – The presumed answer if the user just hits <Enter>. It must be “yes” (the default), “no” or None (meaning an answer is required of the user).

The “answer” return value is one of “yes” or “no”.

Code borrowed from cookiecutter https://github.com/audreyr/cookiecutter/blob/master/cookiecutter/prompt.py

djangocms_installer.utils.**supported_versions**(*django*, *cms*)
Convert numeric and literal version information to numeric format

Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index