{{project_name}} Documentation

Release 0.1.0

noxan

September 03, 2014

Contents

Kick-start your django development with this project template.

Project website: https://github.com/byteweaver/django-genesis

Issue tracker: https://github.com/byteweaver/django-genesis/issues

CHAPTER 1

Key features

- support for multiple targets (e.g. development, production, test)
- requirement tracking and management
- easy deployment to server
- integrated into development process (github)

CHAPTER 2

Abstract

This project template solves every day problems with django and provides some useful additional features. It is designed to be used with the latest stable django (1.7) and python (3.4) version. Furthermore it is ready to be deployed on a Debian GNU/Linux based system within no time and tries to simplify the deployment process.

CHAPTER 3

Usage

```
$ django-admin.py startproject --template=https://github.com/byteweaver/django-genesis/archive/maste
```

CHAPTER 4

Documentation

The following sections contain a detailed documentation on the project template.

4.1 Project template

4.1.1 Project targets

This template extends django to support multiple targets (equivalent to environments) for settings and requirements. This allows us to maintain the source code for various platforms and environments within the same repository under version control without any troubles.

The default targets included in this template are `development`, `production` and `test`. All those extend a common set of either requirements or settings which is called `base` and is designed rather as a helper than a standalone target. For sure you are free to add additional targets suiting your personal needs.

Default target

The default target is `development`. If you like to override the default target (e.g. on your web server for deployment) just create a file called `.target` in the project root directory containing the name of the new default target. This `.target` file is excluded from version control.

For example on a Debian GNU/Linux system you can can set the default target to `production` by calling the following command in your project root directory.

```
$ echo 'production' > .target
```

Using different targets

There is also a way to call commands with a different target than the default one. To change the target of the `make` command just change or set the `TARGET` environment variable to the suiting target name. The `Makefile` will then adjust the requirements and settings like defined in the the value of the environment variable.

To call the requirements makefile target for example once in production mode enter the following command in your project root directory.

```
$ make requirements TARGET=production
```

To do the same but keep the target for all following make calls of your session (would be possible to override the target for ever as well) by calling the following command.

```
$ export TARGET=production
```

Technical details

A target is just a set of two files like already mentioned, one for the requirements and one to set or override settings. The filename without the file extension equals the target's name. Those files are located at `/requirements/{{target-name}}.txt` and `/{{project-name}}/settings/{{target-name}}.py`. The requirements target file just contains a new line separated list of PyPi package names. The handling for base PyPi packages is done by our helper scripts. Meanwhile the settings target file has to or should extend from the common base settings. This is achieved by importing those with a simple one liner: `from {{project-name}}.settings.base import *`.

At the moment there are three entry points for the target selection. One is the `Makefile` which also contains the logic itself. The other two are patched into the default `manage.py` and `{{project-name}}/wsgi.py` scripts while the selection logic itself is centralised in the `{{project-name}}/common/utils.py` script.

4.2 Deployment

4.2.1 Introduction

This guide is ment to be used with django projects based on the django-genesis project template on Debian GNU/Linux based distributions. It will be or at least it should be updated to match the lastest stable django release (1.6.5) and the latest stable python release (3.4).

For use in production you could use the database backend you like, but this guide will cover only PostgreSQL databases for now.

4.2.2 System requirements

This guide covers how to deploy your project behind a nginx proxy server with a gunicorn daemon running as Debian system service. All python requirements will be contained in a dedicated virtual environment.

Summary

- Debian GNU/Linux (wheezy)
- nginx
- PostgreSQL

Debian packages

```
$ apt-get install nginx git python3 python3-dev python-virtualenv
```

Nginx configuration

Optional: Think about turning your configuration folder into a git repository to keep track of your changes.

```
$ cd /etc/nginx
$ git init
$ git add -A
$ git commit -m 'initial configuration files'
```

Create directories for nginx root and default site directories:

```
$ mkdir -p /var/www/default
$ chown -R www-data:www-data /var/www
```

Add content for a default site to a subdirectory in the root directory:

```
$ echo '<html>welcome to nginx!</html>' > /var/www/default/index.html
$ chown www-data:www-data /var/www/default/index.html
```

OR copy content of debian default file:

```
$ cp /usr/share/nginx/www/index.html /var/www/default/
$ chown www-data:www-data /var/www/default/index.html
```

and update nginx configuration to use this directory for default site:

```
$ sed - i 's#/usr/share/nginx/www#/var/www/default#g' /etc/nginx/sites-enabled/default
```

Change ownership of all files contained in `/var/www` always to `www-data`.

PostgreSQL

The desired database backend for production usage.

4.2.3 Project deployment

Github deploy key

Create a ssh keypair for deployment and save it to the `/var/www/.ssh/{{project_name}}/` directory. Change ownership of the directory and key files to `www-data` and grant access privileges only to the `www-data` user (600).

Then add the public key as deploy key to your project on github. You find the `deploy key` page as a tab below the settings page. Click on the `add deploy key` button and paste the content of your public key file.

SSH configuration

To deploy multiple projects for the same user we have to add a ssh configuration to manage multiple deploy keys. The configuration file is stored at `/var/www/.ssh/config`.

```
Host {{project_name}}.github.com
HostName github.com
User git
IdentityFile /var/www/.ssh/{{project_name}}/id_rsa
```

Clone project source code

```
$ cd /var/www
$ git clone {{project_name}}.github.com:{{project_user}}/{{project_name}} {{project_name}}
```

CHAPTER 5

Indices and tables

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