django-project-skeleton Documentation

Release 1.0

Mischback

Feb 05, 2018

Contents

django-project-skeleton is my skeleton for Django projects. It provides a directory structure for Django projects during development and deployment. This structure is based on research and own experience of developing Django apps.

Please note: This is *my* skeleton and is developed to fit my very own needs for new Django projects. Please feel free to modify it to your own requirements but be aware that no changes will be made, that **I** do not consider usefull.

Additional note: As of this writing, Django 1.7 is used. So I can only guarantee that this is working with this version.

CHAPTER 1

Notable Features

- prepared directory structure
- modular settings with sane default values
- prepared sample configuration for *Apache2*-deployment with *mod_wsgi*
- including `.gitignore`-files to help getting started with *Git*

CHAPTER 2

Contents

2.1 Quickstart

I assume you know what you are doing, so let's just do it:

```
$ django-admin startproject --template=https://github.com/Mischback/django-project-
↪skeleton/archive/master.zip [projectname]
```

Your project will look like this:

```
[projectname]/
├── [projectname]/
│   ├── __init__.py
│   ├── settings/
│   │   ├── common.py
│   │   ├── dev.py
│   │   ├── djangodefault.py
│   │   ├── __init__.py
│   │   └── production.py
│   ├── urls.py
│   └── wsgi.py
├── apps/
│   └── __init__.py
├── configs/
│   ├── apache2_vhost.sample
│   └── README
├── doc/
│   ├── Makefile
│   └── source/
│       └── *snap*
├── manage.py
├── README.rst
├── run/
│   ├── media/
│   │   └── README
```

```
    ├── README
    └── static/
        └── README
├── static/
    └── README
└── templates/
    └── README
```

See *Project Structure* for a detailled description of this layout.

2.2 Project Structure

The *normal* Django workflow, as it is described in the official Django tutorial starts a project with the command:

```
$ django-admin startproject [projectname]
```

Your project will look like this:

```
[projectname]/
├── [projectname]/
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── manage.py
```

However, the `startproject`-command takes an optional argument `template` which can be used to specify a project template to be used for project creation (see Django documentation).

The `template`-argument works with paths on your local machine, but also supports URLs. So you can easily fetch this skeleton from **GitHub** using this command:

```
$ django-admin startproject --template=https://github.com/Mischback/django-project-
↪skeleton/archive/master.zip [projectname]
```

Your project will look like this:

```
[projectname]/                  <- project root
├── [projectname]/              <- Django root
    ├── __init__.py
    ├── settings/
        ├── common.py
        ├── dev.py
        ├── djangodefault.py
        ├── __init__.py
        └── production.py
    ├── urls.py
    └── wsgi.py
├── apps/
    └── __init__.py
├── configs/
    ├── apache2_vhost.sample
    └── README
├── doc/
    ├── Makefile
    └── source/
```

```
│       └── *snap*
├── manage.py
├── README.rst
├── run/
│   ├── media/
│   │   └── README
│   ├── README
│   └── static/
│       └── README
├── static/
│   └── README
└── templates/
    └── README
```

2.2.1 Django Root

```
[projectname]/                  <- project root
├── [projectname]/              <- Django root
│   ├── __init__.py
│   ├── settings/
│   │   ├── common.py
│   │   ├── dev.py
│   │   ├── djangodefault.py
│   │   ├── __init__.py
│   │   └── production.py
│   ├── urls.py
│   └── wsgi.py
└── *snap*
```

The Django root directory will be named according to the project name you specified in `django-admin startproject [projectname]`. This directory is the project's connection with Django.

`[projectname]/settings/` Instead of a plain *settings*-file, the configuration is split into several files in this Python module. For an in-depth documentation of these settings see *Settings*.

`[projectname]/urls.py` The root URL configuration of the project. The only configured set of urls is the admin-application. For background information see The Django Book Chapter 3 and The Django Book Chapter 8.

`[projectname]/wsgi.py` Deploying Django makes use of WSGI, the Pythonic way of deploying web applications. See the official settings documentation on WSGI for more details. The default WSGI-application is modified to use our `settings`-module.

2.2.2 apps/

```
[projectname]/                  <- project root
├── *snap*
├── apps/
│   └── __init__.py
└── *snap*
```

This directory is used for custom applications. You can safely remove this directory, if you do not plan to develop custom applications. Most of a Django project's apps will be installed into the Python path and not be kept in your project root.

2.2.3 configs/

This directory contains configuration files for deployment. Now only a configuration file for deployment with **Apache2** and **mod_wsgi** is provided.

```
[projectname]/                  <- project root
├── *snap*
├── configs/
│   ├── apache2_vhost.sample
│   └── README
└── *snap*
```

Please note: It is strongly advised to keep your actual server configuration private. Therefore a `.gitignore`-file is provided, which will only include files ending with the suffix `.sample` into *Git*.

For a brief overview of the `configs/apach2_vhost.sample` refer to *Apache2 Virtual Host Configuration*.

2.2.4 doc/

```
[projectname]/                  <- project root
├── *snap*
├── doc/
│   ├── Makefile
│   └── source/
│       └── *snap*
└── *snap*
```

This directory contains the source files for this documentation.

You can safely remove this directory, if you just want to use the skeleton for your own project.

2.2.5 run/

```
[projectname]/                  <- project root
├── *snap*
├── run/
│   ├── media/
│   │   └── README
│   ├── README
│   └── static/
│       └── README
└── *snap*
```

This directory contains necessary files for running Django. All these files may contain sensible or useless information, so you will not want to keep this files in version control. A `.gitignore`-file is prepared.

This directory will contain the SQLite database file (if you keep the provided `dev`-settings) and the *SECRET_KEY* of Django. For a detailled explanation see *Settings*.

`run/media/` Django uses a special folder to store user-provided files (uploads). In the settings-module of this skeleton this directory is set as `MEDIA_ROOT`.

`run/static/` Similar to media files, all static assets (i.e. stylesheets, javascript files, images) are served from a special directory.

2.2.6 static/ and templates/

```
[projectname]/                  <- project root
├── *snap*
├── static/
│   └── README
└── templates/
    └── README
```

These directories are used for project wide files, meaning project wide static assets and templates.

static/ This directory is used to provide our project wide static assets. Please refer to the Django documentation for more details. *Settings* documents the `STATICFILES_DIRS`-setting.

templates/ This directory is used to provide our project wide templates. *Settings* documents the `TEMPLATE_DIRS`-setting.

2.3 Settings

2.3.1 common.py

This file contains settings which are shared between development- and production-settings. The provide sane defaults for developing and a solid base for production settings.

Path Configuration

DJANGO_ROOT Absolute path of the projects Django directory

PROJECT_ROOT Absolute path of the project directory

SITE_NAME The name of our project

STATIC_ROOT The directory to collect static files into. It will be set to `[project_root]/run/static`. Please refer to the official settings documentation on STATIC_ROOT and this howto on static files.

MEDIA_ROOT The directory for user-uploaded files. It will be set to `[project_root]/run/media`. Please refer to the official settings documentation on MEDIA_ROOT.

STATICFILES_DIRS Django will look in these locations for additional static assets to collect. Our settings module adds `[project_root]/static` to the list. See the official settings documentation on STATICFILES_DIRS for more details.

TEMPLATE_DIRS Django will look in these locations for additional templates. Our settings module adds `[project_root]/templates`. See the official settings documentation on TEMPLATE_DIRS for more details.

Application Configuration

DEFAULT_APPS These are the default apps of `django-admin startproject`. Please note that this is no official setting. Django operates with `INSTALLED_APPS`, which will be set in *dev.py*.

MIDDLEWARE_CLASSES These are the default middleware classes, directly taken from the default settings created by `django-admin startproject`. See the official settings documentation on MIDDLEWARE_CLASSES for more details.

TEMPLATE_CONTEXT_PROCESSORS This setting is not included in the settings of `django-admin startproject` but is added using the default values. See the official settings documentation on TEMPLATE_CONTEXT_PROCESSORS for more details.

Security Configuration

SECRET_FILE Django uses a `SECRET_KEY` for security purposes. As you can clearly see, this is a very sensitive information. We will store this key in a file. This file's location is set up here. Default value is `[project_root]/run/SECRET.key`.

ADMINS You will have to fill this setting yourself, please refer to official settings documentation on ADMINS.

MANAGERS You will have to fill this setting yourself, please refer to official settings documentation on MANAGERS.

Django Running Configuration

WSGI_APPLICATION This setting determines the path to the WSGI-application. We'll use the default one, so this setting is set to `[project_name].wsgi.application`.

ROOT_URLCONF Determines the root URLconf. Set to `[project_name].urls`. See official settings documentation on ROOT_URLCONF.

SITE_ID A unique ID of the site. See official settings documentation on SITE_ID.

STATIC_URL Determines, under which URL static files are served. You will want to adjust this in a production scenario. Our default value is `/static/`. See official settings documentation on STATIC_URL.

MEDIA_URL Determines, under which URL media files are served. You will want to adjust this in a production scenario. Our default value is `/media/`. See official settings documentation on MEDIA_URL.

Debug Configuration

DEBUG Activates debugging. In this file, this is set to `False`, because these are our common settings, which are shared between all configurations. We just want debugging while we are developing, so debugging will be activated in *dev.py*. See official settings documentation on DEBUG for additional information.

TEMPLATE_DEBUG I have never really understood why `DEBUG` is seperated from `TEMPLATE_DEBUG`. So let's just keep them in sync.

Internationalization

LANGUAGE_CODE Sets the language of this project. See official settings documentation on LANGUAGE_CODE.

TIME_ZONE Sets the time zone of this project. See official settings documentation on TIME_ZONE.

USE_I18N Activates Django's translation system. See official settings documentation on USE_I18N.

USE_L10N Activates Django's localization engine. See official settings documentation on USE_L10N.

USE_TZ Make datetimes timezone aware. See official settings documentation on USE_TZ.

2.3.2 dev.py

This file contains development settings.

Debug Configuration

DEBUG We are developing, so activate debugging.

TEMPLATE_DEBUG See my above nerdrage about template debugging vs. debugging. Keep them in sync.

Database Configuration

DATABASES I use SQLite for development. The database file will be created in `[project_root]/run/dev.sqlite3`.

Application Configuration

INSTALLED_APPS We have set the default apps. Now we build the (required) `INSTALLED_APPS`-setting by using `DEFAULT_APPS` and add any app we need for development.

2.3.3 production.py

This file contains all production settings. Please note, that the current setup leaves this empty and simply imports the dev-settings. This is done, because we have adjusted `manage.py` and `[project_root]/wsgi.py` to use the production settings.

2.3.4 djangodefault.py

This are the saved settings from `django-admin startproject`. We just keep them for completeness, these settings are not actually used.

2.4 Apache2 Virtual Host Configuration

This is an Apache2 configuration file for name based virtual hosting.

As you can see in the following listing, there are several placeholders, that must be filled to make this work.

2.4.1 Usage

As you may notice, there are three different types of placeholders.

[[placeholder_name]] These placeholders must be filled manually. Most noticable is line 4, where you **must** set the server name.

```
ServerName [[SERVER_NAME]]
```

${placeholder_name} These placeholders are filled by Apache itsself. Only mess with them, if you do exactly know what you are doing.

{{ placeholder_name }} These placeholders do look familiar, don't they? These are Django templatetags. You may fill them manually (please refer to the provided resources in the comments), but you can Django let them fill them for you during project creation. This will render the file through Django's template engine and fill these placeholders:

```
$ django-admin startproject --template=/path/to/template --name apache2_vhost.
↪sample
```

2.4.2 Concept

This will set up a name based virtual host that uses *mod_wsgi* to interact with Django.

It will serve static- and media-files from the default locations set in `settings/common.py`. This is not a production-setting, but is well suited for development purposes.

Line 10: `Alias /static/ {{ project_directory }}/run/static` Serve static files from `STATIC_ROOT` under `STATIC_URL`. Note lines 36 - 40, where the directory is made accessible for Apache.

Line 15: `Alias /media/ {{ project_directory }}/run/media` Serve media files from `MEDIA_ROOT` under `MEDIA_URL`. Note lines 45 - 49, where the directory is made accessible for Apache.

The dynamic Django content is served using the *WSGI-application*. Apache2 will use *mod_wsgi* in Daemon-mode. This is in fact the preferred way of deploying Django with Apache2, so you will not have to mess with these settings.

Line 18: `WSGIScriptAlias / {{ project_directory }}/{{ project_name }}/wsgi.py` This must be set to the absolute filesystem path to the *WSGI-application*.

Line 27: `WSGIDaemonProcess ...` This sets the name of the daemon process. Using Django's template engine, this will be set to the name of your project. Please notice the `python-path`-parameter. It is prepared to a virtualenv-setup, but frankly, it must contain the *project directory* and the path to Python's *site-packages*.

Line 31: `WSGIProcessGroup ...` Specifies a distinct name for the daemon process's group.

2.4.3 Source

```
1  <VirtualHost *:80>
2      # This is name based virtual hosting. So place an appropriate server name
3      #   here. Example: django.devsrv.local
4      ServerName  [[SERVER_NAME]]
5      ServerAdmin webmaster@localhost
6
7      # This alias makes serving static files possible.
8      #   Please note, that this is geared to our settings/common.py
9      #   In production environment, you will propably adjust this!
10     Alias /static/  {{ project_directory }}/run/static/
11
12     # This alias makes serving media files possible.
13     #   Please note, that this is geared to our settings/common.py
14     #   In production environment, you will propably adjust this!
15     Alias /media/  {{ project_directory }}/run/media/
16
17     # Insert the full path to the wsgi.py-file here
18     WSGIScriptAlias /   {{ project_directory }}/{{ project_name }}/wsgi.py
19
20     # PROCESS_NAME specifies a distinct name of this process
21     #   see: https://code.google.com/p/modwsgi/wiki/ConfigurationDirectives
↪#WSGIDaemonProcess
22     # PATH/TO/PROJECT_ROOT is the full path to your project's root directory,
23     #   containing your project files
24     # PATH/TO/VIRTUALENV/ROOT: If you are using a virtualenv specify the full
```

```
    #   path to its directory.
    #   Generally you must specify the path to Python's site-packages.
    WSGIDaemonProcess   {{ project_name }}  python-path={{ project_directory }}:{{ project_directory }}/../lib/python2.7/site-packages

    # PROCESS_GROUP specifies a distinct name for the process group
    #   see: https://code.google.com/p/modwsgi/wiki/ConfigurationDirectives#WSGIProcessGroup
    WSGIProcessGroup    {{ project_name }}

    # Serving static files from this directory
    #   Please note, that this is geared to our settings/common.py
    #   In production environment, you will propably adjust this!
    <Directory {{ project_directory }}/run/static>
        Options -Indexes
        Order deny,allow
        Allow from all
    </Directory>

    # Serving media files from this directory
    #   Please note, that this is geared to our settings/common.py
    #   In production environment, you will propably adjust this!
    <Directory {{ project_directory }}/run/media>
        Options -Indexes
        Order deny,allow
        Allow from all
    </Directory>

    LogLevel warn

    # PROJECT_NAME is used to seperate the log files of this application
    ErrorLog    ${APACHE_LOG_DIR}/{{ project_name }}_error.log
    CustomLog   ${APACHE_LOG_DIR}/{{ project_name }}_access.log combined
</VirtualHost>
```