Doorstep Documentation

Release dev

Faraz Masood Khan

Mar 27, 2018

Contents

Doorstep is powerful e-commerce solution built on top of Django. This documentation will help you development e-commerce site using Doorstep and take leverage of all the goodies that Django & Doorstep provides.

You can report bugs and discuss features on the issues page.

CHAPTER 1

First steps

Do you know Django? than this is the place to start!

- *Overview* of Doorstep.
- *Installation* guide for Doorstep.
- *Start new project* for e-commerce site.

Contents:

1.1 Overview

Doorstep is open source e-commerce solution, simplicity in designed is to thrive sales, which are reaching to be production ready. It is built on top of **Django_** Web-development framework for Python.

1.1.1 Doorstep Apps

Doorstep is set of Django apps similar to builtin apps like session, auth, admin and etc, each app is design to serve specific purpose.

- `doorstep`: core app for base classes of views and templates and hold all urls
- `doorstep.geo`: contains models for country, state & addresses
- `doorstep.pages`: to serve static pages for about, contact and policy
- `doorstep.accounts`: extends Django auth model and also provide abstract classes
- `doorstep.catalog`: products catalog and listings
- `doorstep.sales`: order processing
- `doorstep.financial`: currency rate and conversion
- `doorstep.payments`: payment gateways like PayPal & Stripe

1.1.2 Built With

- **Django_**: web development framework for python, we utilizes full stack.
- LESS: styling totally done in LESS, a preprocessor for CSS.
- Django-Pipeline: we use django-pipeline to compile & compress LESS and also compress Javascript before deployment.
- PostgreSQL: I would recommend to use PostgreSQL for production, but project intended to support all databases that Django supports. SQLite is good alternative for small sites, let say 1000 orders per week won't break a sweat, see limit for SQLite for more details, but it has lacks good tools for administration

1.2 Installation

Before you can start with your e-commerce project, lets install Doorstep.

1.2.1 Virtual environment

Create a new virtual environment for Doorstep, its isolated Python environment which are more practical than installing Doorstep systemwide. They also allow installing packages without root privileges, you may create separate virtualenv for each of your e-commerce site.

```
$ virtualenv doorstep_env && source doorstep_env/bin/activate
```

1.2.2 Install Doorstep

Recommend way to install Doorstep is via pip, it will be easy to upgrade to latest version. Alternatively you can download repository and install with via setuptools `python setup.py install`

```
$ pip install --upgrade git+https://github.com/mysteryjeans/doorstep.git#egg=Doorstep
```

1.2.3 Install LESS & Yuglify

Install LESS & Yuglify nodejs packages, which will be used by django-pipeline for javascript and CSS processing, nodejs setup also include npm package manager.

```
$ npm install -g less yuglify
```

1.3 Project

I'll assume you have *Doorstep installed* already. You can tell which version by running the following command:

```
$ python -c "import doorstep; print(doorstep.get_version())"
```

If Doorstep is installed, you should see the version of your installation. If it isn't, you'll get an error telling "No module named doorstep".

1.3.1 Start new project

If you haven't develop with Django before, this documentation doesn't cover you for Django development. Doorstep itself follows Django philosophy and uses, extends or even copies it where ever possible.

Start your own e-commerce project using `doorstep-admin.py`, it simple wrapper around Django's own `django-admin.py`:

```
$ doorstep-admin.py startproject ecomstore
```

Now that your own site is created, lets change directory to ecomstore & create database tables by running following command. This will create all tables required by Django & Doorstep. Default database is SQLite which is a good starting point, you can later switch to your favorite databases that are supported by Django.

```
$ python manage.py migrate
```

1.3.2 Run your site

Let's run the builtin Django development server and verify by visiting http://127.0.0.1:8000, if all works well you will see web page with not products & listings. If you see error compiling CSS or Javascript than head over to *installation* and install LESS & Yuglify nodejs packages.

```
$ python manage.py runserver
```

1.3.3 Create site admin

Let's create site admin or in Django term superuser. There are two ways to create site admin, by Django's builtin command or the first user that register to the site will automatically becomes site admin.

```
$ python manage.py createsuperuser
```

1.4 Configuration

Doorstep also requires settings to be defined in `settings.py`, you have probably *create new project* for site which should have following extra settings beside standard Django settings parameters:

```
################## Doorstep ########################
# Doorstep e-commerce settings for Django project
# Customize these settings only if you know
from doorstep.settings import *
INSTALLED_APPS += DOORSALE_APPS
```

This code is effectively importing all the settings required by Doorstep into your project's `settings.py`. However Doorstep apps doesn't replace your project's `INSTALLED_APPS`, therefore `DOORSALE_APPS` must be added to your projects installed apps explicitly.

Override Settings!

All settings variable should be defined after Doorstep's default imported in `settings.py`

1.4.1 User Model

Doorstep extends `django.contrib.auth` by driving from Django's auth user model abstract classes and defined it's own user auth model in `settings.py`. Where `accounts.User` is compose of app name **accounts** that contains auth user model **User**.

```
AUTH_USER_MODEL = 'accounts.User'
```

You can extend Doorstep's auth user model in similar way by creating your own authentication app `myauth` and drive `MyUser` user model from abstract classes provided in `doorstep.accounts`.

```
from django.db import models
from doorstep.accounts.models import AbstractUser

class MyUser(AbstractUser):
    mail_digest = models.BooleanField(default=True)
```

Lastly override `AUTH_USER_MODEL` in project `settings.py` after Doorstep settings. Remember to use `get_user_model` from `django.contrib.auth` instead of directly references user auth model class.

```
AUTH_USER_MODEL = 'myauth.MyUser'
```

1.4.2 Login Auth

Authenticate is handle by accounts apps, you don't need to override login url for your custom `AUTH_USER_MODEL`. If you want to implement your custom login page, let say for integration with other account services like Google & Facebook. You can simple implement your custom login and override login url to hit your own View

```
LOGIN_URL = '/accounts/login/'
```

1.4.3 Login Redirect

After authentication if return URL is exists in `next` parameter in query string then user will automatically redirect URL defined login redirect url

```
LOGIN_REDIRECT_URL = '/'
```