Extending an existing Django Custom App

Eleanor Cassady

March 20, 2025

Contents

1 Installation

At this point in time, our apps are deployed via heroku. This may change in the future and this document updated accordingly.

1.1 Clone from heroku

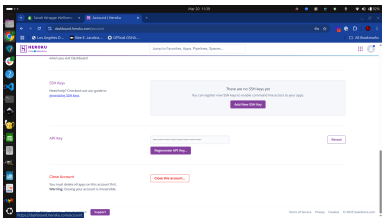
Locate the app in the sunnyroad -dev section on the company heroku, cd to an appropriate directory and run *heroku git:clone -a <app-name>*. cd into this directory

1.2 Add Github Remote

Locate the app in github, and add it as the origin: *git remote add origin git@github.com:Chelsea-And-Rachel/<repository>.git*

1.3 Obtain heroku API key

From your heroku dashboard, copy or create a new API key



1.4 Pull Configuration from Heroku

Heroku is configured for everything you will need to develop the app locally, except for the current OAuth key. Run the following command:

./bin/pull_configuration –heroku_key <heroku key> –app <app name>

1.5 Initialize and enter virtual environment

In the app directory run *pipenv install && pipenv shell*

1.6 Retrieve the current OAuth Token for local development

run *./bin/get_token.py* to return the currenrt OAuth token. For local development you can apply this in one of two ways:

For the current session: *export SHOPIFY_TOKEN=‘./bin/get_token.py‘*

For all sessions: *echo TEST_TOKEN=‘./bin/get_token.py‘ >> .env*

1.7 Test Local environment

run *./manage.py runserver* to run the app locally, usually available at 127.0.0.1:8000

2 Support Scripts

A small number of support scripts have been created to aid in local development. They live in *bin*

2.1 configure.py

This is only for the initial configuration of an entirely new app and is covered in another document.

2.2 get_token.py

Retrieves the current OAuth token of the installed app

2.3 mysql.sh

Launches the mysql client for the MySQL database configured for the app

2.4 pull_configuration.py

Pulls the current heroku configuration for local development

2.4.1 Arguments

- –heroku_key the heroku API key attached to your heroku account.
- –app the heroku app to pull configuration from

2.5 update-heroku.py

Updates the heroku configuration from the local *.env* file. The use case for this script is incredibly rare and should be used with the greatest caution: *heroku config:set* is preferred.

2.5.1 Arguments

- –heroku_key the heroku API key attached to your heroku account.
- –app the heroku app to pull configuration from

2.6 webhooks.py

Test shopify webhooks[1] locally or remotely with a preconfigured JSON payload specific to the webhook type.

2.6.1 Arguments

–list flag to list all currently configured webhooks. displays raw GraphQL return.
–trigger trigger a webhook
–local trigger the webhook in the development environment
–path the mapped path of the webhook e.g. *webhooks/order-created*

2.6.2 Payload

A payload in the format corresponding to the topic[2] attached the webhook. This is fed into the script from STDIN

2.6.3 Example

cat payload.json | ./bin/webhooks.py –trigger –local –path webhooks/order-created

3 App Structure

3.1 The home app

Located in the */home/* directory, this is the Django app that contains all the code that should be modified. Several common applications—creating and configuring webhooks; Shopify Discount Functions and Cart Transform Functions—are built into the app skeleton. This code is located in *home/cnr* in legacy apps and *home/srd* in apps developed after the merger and is off-limits unless absolutely necessary, done with great care and if applicable to future development backfilled into the skeleton.

3.2 The Base Django app

located in the */shopify_django_app/* directory, this is the core of the Django app, with global configuration. This is only rarely to be updated.

[1] other services with webhoooks registered to the app can be listened for as well.
[2] https://shopify.dev/docs/api/webhooks?reference=toml

3.3 The Shopify Authorization app

located in the */shopify_app/* directory, this handles the Shopify authorization flow. Barring changes to that flow, it is absolutely not to be touched.

3.4 The Project Folder

Located in /home/project/ directory, this is where project-specific code lives, in three main files:

3.4.1 /home/project/urls.py

This maps paths to corresponding functions. Standard MVC stuff.

3.4.2 /home/project/views.py

Contains the functions that will process all incoming requests not otherwise handled by the base install. These functions operate primarily with two decorators:

- *@shopify_login_required* These functions require an authenticated shopify session.
- *@csrf_exempt* This decorator applies to webhooks, in which the authentication is verified in request headers and CORS-applicaple requests from the shopify front-end.

3.4.3 /home/project/webhooks,py

Contains webhook handling classes inheriting from *WebhookBase*.

3.5 Shopify Extensions

These live in the */shopify-extensions/* directory, which contains a stub shopify project in which extensions can be created, tested and deployed. The ins and outs of shopify extension development are beyond the scope of this document. *nota bene*: not all function types are available to clients not on Shopify Plus plans.

4 Miscellany

4.1 Discount, Webhook and Cart Transform CRUD Javascripts

These are evolving, but handle CRUD operations for these use cases. Currently, configuration in functions is embedded JSON in the relevant template, but this will soon be replaced with a metaobject-based solution requiring no template editing.

4.2 Topics not covered

Should any clarification on this subject be required, please direct all requests to the author of this guide, *ellie@sunnyroad.co*