django-chartwerk Documentation

Release 0.0.5

Jon McClure and The Dallas Morning News

Jul 02, 2017

Contents:

Chartwerk is an application for developing data visualizations and publishing them as embeddable, flat pages to Amazon S3.

CHAPTER 1

Why Chartwerk?

Like many other chart builders, Chartwerk provides an interface for non-coders to easily create interactive and static charts. However, you may find, like we did, that most chart makers are set-and-forget systems that aren't well designed to grow with the needs of your team.

Chartwerk was designed to be a more collaborative tool between coders and non-coders. It lets developers easily build and modify charts on the fly directly alongside users by exposing a robust internal API that translates tabular data into discrete dataviz properties.

Because chart templates in Chartwerk are arbitrary functions written to consume Chartwerk's API, developers have complete control of the logic used to draw charts and the freedom to use any third-party libraries they like.

In the newsroom, Chartwerk helps us develop dataviz quickly in response to the needs of beat reporters and scale our development time multiplied by every chart our reporters build from the templates we create.

That said, Chartwerk may not be the best choice among all other chart builders for your team if you don't have at least one developer to help build up your chart template set.

What's in it?

Chartwerk actually consists of two applications:

1. A backend app that maintains RESTFUL endpoints for charts and chart templates, serves navigational pages for users to select the type of chart they'd like to build and handles logic for user accounts and for "baking" charts to S3 or another flat storage service.
2. A front-end app to create and manipulate charts and chart templates before saving them to the backend.

Django-chartwerk represents the former. You can find the latter at chartwerk-editor.

Note: Chartwerk-editor is **the heart of Chartwerk**. It is the app users interact with to create charts and chart templates.

Django-chartwerk represents a deployment package for the editor. (It actually *includes* the editor within itself.)

Most of the information you'll need to understand how to use Chartwerk, to interact with Chartwerk's internal API, to build chart templates as well as the logic behind Chartwerk's workflow is in chartwerk-editor's documentation. Read it here.

Use these docs to deploy Chartwerk within a pluggable Django app.

CHAPTER 2

Installing

Assumptions

1. django-chartwerk is written to save charts to Amazon Web Service's Simple Storage Service (S3). We assume that's your plan, too.
2. django-chartwerk uses Django's JSONField field, therefore, the app requires a PostgreSQL database.

Note: If you're not already using PostgreSQL in a project you'd like to add django-chartwerk to, you can separate django-chartwerk's database from your default database by using a custom router, as outlined in the Django documentation.

Quickstart

1. Install django-chartwerk using pip.

```
$ pip install django-chartwerk
```

2. Add chartwerk's dependencies and the minimum configuration variables.

```
# project/settings.py

INSTALLED_APPS = [
    # ...
    'django.contrib.humanize',
    'rest_framework',
    'chartwerk',
]

CHARTWERK_DOMAIN = 'https://yourapp.com'
CHARTWERK_EMBED_SCRIPT = 'https://yourapp.com/static/wherever/js/embed_v1.js'
```

```
CHARTWERK_AWS_BUCKET = 'chartwerk'
CHARTWERK_AWS_ACCESS_KEY_ID = 'YOUR_ACCESS_KEY'
CHARTWERK_AWS_SECRET_ACCESS_KEY = 'YOUR_SECRET_KEY'
```

Note: Just trying out Chartwerk locally? Set the above **CHARTWERK_** variables to gibberish. They're only needed when you start publishing charts but will throw errors if they aren't set.

3. Add chartwerk to your project's *urls.py*.

```
# project/urls.py

urlpatterns = [
  # ...
  url(r'^chartwerk/', include('chartwerk.urls')),
]
```

4. Chartwerk uses Celery to process some tasks asynchronously. Read "First steps with Django" to see how to setup a Celery app in your project. Here is a configuration you can also use to start:

```
# project/celery.py
import os

from celery import Celery
from django.conf import settings

os.environ.setdefault('DJANGO_SETTINGS_MODULE', '<your project>.settings')

app = Celery('chartwerk')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.conf.update(
  task_serializer='json'
)
# Use synchronous tasks in local dev
if settings.DEBUG:
  app.conf.update(task_always_eager=True)
app.autodiscover_tasks(lambda: settings.INSTALLED_APPS, related_name='celery')


# project/__init__.py
from .celery import app as celery_app

__all__ = ['celery_app']
```

5. Run migrations and start the dev server!

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

CHAPTER 3

Configuring

Chartwerk allows you to set a number of configuration options. Some add additional features to the app.

App settings

Listing 3.1: Default settings

```
CHARTWERK_AUTH_DECORATOR = "django.contrib.auth.decorators.login_required"
CHARTWERK_API_PERMISSION_CLASS = "rest_framework.permissions.IsAuthenticatedOrReadOnly
↪"
CHARTWERK_COLOR_SCHEMES = {} # Uses default color scheme in chartwerk-editor
```

CHARTWERK_AUTH_DECORATOR

String module path to a decorator that should be applied to Chartwerk views to authenticate users.

> **Warning:** This decorator is not applied to views if DEBUG is true in your settings.

CHARTWERK_API_PERMISSION_CLASS

String module path to a valid Django REST permission class that should be applied to the browsable API viewsets.

CHARTWERK_COLOR_SCHEMES

Set this variable in your project settings to declare a default set of color schemes your users can select for chart elements. The schemes must be organized by type as a dictionary with keys *categorical*, *sequential* and *diverging*. Name each color scheme and then provide a list of hexadecimal color codes. For example:

```
# settings.py

CHARTWERK_COLOR_SCHEMES = {
  'categorical': {
      'default': [
          '#AAAAAA',
          '#BBB'
          # etc.
      ],
  }
  'sequential': {
      'reds': [
          '#FF0000',
          '#8B0000',
          # etc.
      ],
      'blues': [
          '#0000FF,
          '#000080',
          # etc.
      ]
  },
  'diverging': {
      'redBlue': [
          '#FF0000',
          '#0000FF',
          # etc.
      ]
  }
}
```

AWS

Listing 3.2: Default settings

```
CHARTWERK_AWS_ACCESS_KEY_ID = None  # Required
CHARTWERK_AWS_SECRET_ACCESS_KEY = None  # Required
CHARTWERK_AWS_BUCKET = None  # Required
CHARTWERK_AWS_PATH = "charts"
CHARTWERK_CACHE_HEADER = "max-age=300"
CHARTWERK_DOMAIN = None  # Required
CHARTWERK_EMBED_SCRIPT = None  # Required
CHARTWERK_JQUERY = "https://code.jquery.com/jquery-3.2.1.slim.min.js"
```

CHARTWERK_AWS_ACCESS_KEY_ID

Amazon Web Services access key ID. **Required.**

CHARTWERK_AWS_SECRET_ACCESS_KEY

AWS secret access key. **Required.**

CHARTWERK_AWS_BUCKET

AWS S3 bucket name to publish charts to. **Required.**

CHARTWERK_AWS_PATH

Path within your S3 bucket to append to object keys before publishing.

CHARTWERK_CACHE_HEADER

Cache header to add to chart files when published to S3.

CHARTWERK_DOMAIN

The domain of the app running Chartwerk. For example, your app may be hosted at *http://myapp.mydomain.com*.

CHARTWERK_EMBED_SCRIPT

Absolute URL to your custom script for embedding Chartwerk charts in your CMS.

CHARTWERK_JQUERY

URL to jQuery version you want to include in baked-out charts.

GitHub

Django-chartwerk can commit your chart templates to a GitHub repository for safe keeping.

Listing 3.3: Default settings

```
CHARTWERK_GITHUB_ORG = None
CHARTWERK_GITHUB_REPO = "chartwerk_chart-templates"
CHARTWERK_GITHUB_USER = None
CHARTWERK_GITHUB_PASSWORD = None
CHARTWERK_GITHUB_TOKEN = None
```

CHARTWERK_GITHUB_ORG

To keep templates in a repo under a GitHub organization, set this variable to the GitHub org name.

CHARTWERK_GITHUB_REPO

The name of the repo to save chart templates to.

CHARTWERK_GITHUB_USER

GitHub username to access GitHub API.

Note: We recommend you use a personal access token instead of setting your username and password in these settings.

CHARTWERK_GITHUB_PASSWORD

Password for your GitHub username.

CHARTWERK_GITHUB_TOKEN

GitHub personal access token with rights to edit private repositories.

Slack

Chartwerk can send notifications to a Slack channel whenever a new chart is created.

Listing 3.4: Default settings

```
CHARTWERK_SLACK_CHANNEL = "#chartwerk"
CHARTWERK_SLACK_TOKEN = None
```

CHARTWERK_SLACK_CHANNEL

Name of the Slack channel to post notifications to.

CHARTWERK_SLACK_TOKEN

A Slack API token.

oEmbed

Chartwerk can act as an oEmbed provider, returning embeddable charts using an oEmbed endpoint at `api/oembed`.

Listing 3.5: Default settings

```
CHARTWERK_OEMBED = False
CHARTWERK_OEMBED_EXTRA_PATTERNS = []
```

CHARTWERK_OEMBED

Set to `True` to have the oEmbed endpoint returned in the API's context object.

CHARTWERK_OEMBED_EXTRA_PATTERNS

If you'd like the oEmbed endpoint to support any additional URL patterns, provide them here. This can be useful if, for example, you alter your root URL configuration and all of the chart URLs change. Each pattern should be provided as a regular expression, with named capture groups that can be used to lookup charts. For example:

```
# settings.py

CHARTWERK_OEMBED_EXTRA_PATTERNS = (
  r'^old-chartwerk/chart/(?P<slug>[-\w]+)/$',
)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search