django-chartwerk Documentation

Release 0.5.19

Jon McClure and The Dallas Morning News

Oct 05, 2018

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.

CHAPTER 2

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 (demo).

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 latest version of 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 the docs here.

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

2.1 Free chart templates

Django-chartwerk includes a number of chart templates to get you started using Chartwerk. They are loaded automatically through fixtures when you first migrate the application. You get:

- Bar chart
- Column chart
- Multi-line chart
- Unit chart

- US state choropleth map
- Data table

You'll see many of the concepts described in the chartwerk-editor docs in practice in these templates. Use them as a starting point to build your own or customize them to suit your needs.

CHAPTER 3

Installing

3.1 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 9.4.

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. See "*Using a database router*" for an example.

3.2 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'
```

(continues on next page)

(continued from previous page)

```
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 chartwerk migrations and load free chart templates from fixtures.

```
$ python manage.py migrate chartwerk
$ python manage.py loaddata free_charts
```

6. Start the dev server and enjoy!

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

Note: The default permissions setting for Chartwerk's backend requires that users are authenticated. So in a new project with `DEBUG = true`, an unauthenticated user can access the Editor, but she cannot save a chart to the backend (403 error).

For new projects, then, be sure to create a user account and login before trying to save charts.

3.3 Using a database router

If you'd like to separate django-chartwerk's PostgreSQL database from the database(s) used in the rest of your Django project, you can write and connect a router.

For example:

```
# project/routers.py
class ChartwerkRouter(object):
  def db_for_read(self, model, **hints):
    if model._meta.app_label == 'chartwerk':
      return 'chartwerk'
    else:
      return 'default'

  def db_for_write(self, model, **hints):
    if model._meta.app_label == 'chartwerk':
      return 'chartwerk'
    else:
      return 'default'

  def allow_relation(self, obj1, obj2, **hints):
    if obj1._meta.app_label == 'chartwerk' or obj2._meta.app_label == 'chartwerk':
      return True
    return None

  def allow_migrate(self, db, model):
    if db == 'chartwerk':
      return model._meta.app_label == 'chartwerk'
    elif model._meta.app_label == 'chartwerk':
      return False
    return None
```

Add your router and database in settings.

```
# project/settings.py
import dj_database_url

# Add chartwerk DB to existing DB settings
DATABASES['chartwerk'] = dj_database_url.parse('postgres://...')
DATABASE_ROUTERS = [
  'project.routers.ChartwerkRouter',
]
```

When you separate django-chartwerk's database, you must specify the database explicitly when running migrations to create models and loading fixtures.

```
$ python manage.py migrate chartwerk --database chartwerk
$ python manage.py loaddata free_charts --database chartwerk
```

CHAPTER 4

Configuring

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

4.1 App settings

Listing 1: Default settings

```
CHARTWERK_AUTH_DECORATOR = "django.contrib.auth.decorators.login_required"
CHARTWERK_API_AUTHENTICATION_CLASSES = ("rest_framework.authentication.
↪SessionAuthentication",)
CHARTWERK_API_PERMISSION_CLASSES = ("rest_framework.permissions.
↪IsAuthenticatedOrReadOnly",)
CHARTWERK_COLOR_SCHEMES = {} # Uses default color scheme in chartwerk-editor
```

4.1.1 `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.

4.1.2 `CHARTWERK_API_AUTHENTICATION_CLASSES`

Iterable of string module paths to valid Django REST authentication classes that should be applied to Django REST Framework's browsable API viewsets.

4.1.3 CHARTWERK_API_PERMISSION_CLASSES

Iterable of string module paths to valid Django REST permission classes that should be applied to Django REST Framework's browsable API viewsets.

4.1.4 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.
      ]
  }
}
```

Warning: You should specify a `default` color scheme under the `categorical` key. You can name all other schemes whatever you want.

4.2 AWS

Listing 2: Default settings

```
CHARTWERK_AWS_ACCESS_KEY_ID = None  # Required
CHARTWERK_AWS_SECRET_ACCESS_KEY = None  # Required
CHARTWERK_AWS_BUCKET = None  # Required
```

(continues on next page)

(continued from previous page)

```
CHARTWERK_AWS_PATH = "charts"
CHARTWERK_AWS_REGION = "us-east-1"
CHARTWERK_CACHE_HEADER = "max-age=300"
CHARTWERK_CLOUDFRONT_DISTRIBUTION = None
CHARTWERK_DOMAIN = None  # Required
```

4.2.1 `CHARTWERK_AWS_ACCESS_KEY_ID` (Required)

Amazon Web Services access key ID.

4.2.2 `CHARTWERK_AWS_SECRET_ACCESS_KEY` (Required)

AWS secret access key.

4.2.3 `CHARTWERK_AWS_BUCKET` (Required)

AWS S3 bucket name to publish charts to.

4.2.4 `CHARTWERK_DOMAIN` (Required)

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

4.2.5 `CHARTWERK_AWS_REGION`

Region of your AWS bucket.

4.2.6 `CHARTWERK_AWS_PATH`

Path within your S3 bucket to append to your charts when publishing. For example, setting to `chartwerk/charts` would result in charts published to `chartwerk/charts/<chart_id>.html` in your bucket.

4.2.7 `CHARTWERK_CACHE_HEADER`

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

4.2.8 `CHARTWERK_CLOUDFRONT_DISTRIBUTION`

If you’re using Amazon CloudFront in front of your S3 bucket and would like to create an invalidation whenever charts are updated, add your distribution ID to this setting.

4.3 Compiling JavaScript

4.3.1 CHARTWERK_JS_SUBPROCESS

If you'd like to write your template scripts using modern JavaScript syntax, you can pipe them through a compiler before publishing by specifying arguments to pass to Python's subprocess module.

For example, if you'd like to compile ES2015 syntax using Babel, you could specify subprocess args like this:

```
CHARTWERK_JS_SUBPROCESS = ['npx', 'babel', '--presets=es2015']
```

This option assumes you have already installed the dependencies referenced in your subprocess on your server. Obviously, you can only use CLI compilers with this method.

Note: This method **does not** compile scripts in the Editor. You should use a browser that supports the syntax features you're targeting when you and your users develop charts. JavaScript will be compiled before baking your charts to S3.

4.4 GitHub

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

Listing 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
```

4.4.1 CHARTWERK_GITHUB_ORG

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

4.4.2 CHARTWERK_GITHUB_REPO

The name of the repo to save chart templates to.

4.4.3 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.

4.4.4 CHARTWERK_GITHUB_PASSWORD

Password for your GitHub username.

4.4.5 CHARTWERK_GITHUB_TOKEN

GitHub personal access token with rights to edit private repositories.

4.5 Slack

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

Listing 4: Default settings

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

4.5.1 CHARTWERK_SLACK_CHANNEL

Name of the Slack channel to post notifications to.

4.5.2 CHARTWERK_SLACK_TOKEN

A Slack API token.

4.6 oEmbed

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

Listing 5: Default settings

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

4.6.1 CHARTWERK_OEMBED

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

4.6.2 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]+)/$',
)
```

4.6.3 Configuring an oEmbed integration

Configure your CMS's oEmbed integration to make GET requests to django-chartwerk's oEmbed endpoint at `/api/oembed/`. An example might look like: `https://myapp.com/chartwerk/api/oembed/`.

At minimum, you need to send an encoded URI for the chart you'd like to embed in a `url` query parameter. In django-chartwerk, charts have two canonical URIs:

- `/chart/<chart ID>/`
- `/api/charts/<chart ID>/`

The embed code generator in chartwerk-editor will return the latter to the user when `CHARTWERK_OEMBED = True`.

So an oEmbed request might look like:

`https://myapp.com/chartwerk/api/oembed/?url=https%3A%2F%2Fmyapp.com%2Fchartwerk%2Fchart%2F<chart ID>%2F`

Remember, django-chartwerk will bake out two chart sizes, double and single-wide. Your integration is responsible for passing a user's *preferred* chart size in the oEmbed request as an additional query string parameter, `size={single|double}`.

A response – using the default embed code – may look like this:

```
{
  "version": "1.0",
  "url": "https:\/\/myapp.com\/chartwerk\/chart\/<chart ID>\/",
  "title": "A map",
  "provider_url": "https:\/\/myapp.com\/chartwerk\/",
  "provider_name": "Chartwerk",
  "author_name": "user@email.com",
  "chart_id": "<chart ID>",
  "type": "rich",
  "size": "double",
  "width": 600,
  "height": 494,
  "single_width": 290,
  "single_height": 329,
  "html": "<div id=\"chartwerk_<chart ID>\" class=\"chartwerk\" data-id=\"<chart ID>\
↪" data-dimensions=\"{&quot;double&quot;: {&quot;width&quot;: 600, &quot;height&quot;
↪: 494}, &quot;single&quot;: {&quot;width&quot;: 290, &quot;height&quot;: 329}}\"␣
↪data-size=\"double\" data-src=\"https:\/\/myS3bucket.com\/charts\/chartwerk\/\" ><\/
↪div> <script src=\"https:\/\/myS3bucket.com\/charts\/chartwerk\/embed-script\/v1.js\
↪"><\/script>"
}
```

The `html` property in the response object will be generated using `CHARTWERK_EMBED_TEMPLATE`. Your integration should use it to inject your embed code into your page.

4.7 Embed code

These settings configure the code used to embed your charts in a page. The code is either returned to your users directly in the Editor or sent as part of the oEmbed response object, if oEmbed is configured.

The embed code is responsible for injecting an iframe into a page, setting its source to either the single or double-wide chart and, usually, setting its height, width, margins and float styles. (The default embed code uses Pym.js.)

By templatizing the embed code, django-chartwerk gives you the freedom to write exactly the code you need for your CMS. The settings consist of a template string, which you can write to include any arbitrary HTML, CSS, or JavaScript, and a context object that allows you to render your tempate with context from a chart instance.

Note: These aren't required settings, but the defaults will be generally useless. At minimum, you should change the embed template context, `CHARTWERK_EMBED_TEMPLATE_CONTEXT`.

Listing 6: Default settings

```
CHARTWERK_EMBED_TEMPLATE = """
<div
  id="chartwerk_{{id}}"
  class="chartwerk"
  data-id="{{id}}"
  data-dimensions="{{dimensions|safe}}"
  data-size="{{size}}"
  data-src="{{chart_path}}"
></div>
<script src="{{embed_script}}"></script>
"""

CHARTWERK_EMBED_TEMPLATE_CONTEXT = lambda chart: {
    'chart_path': 'http://www.somesite.com/path/to/charts/',
    'embed_script': '<CHARTWERK_DOMAIN>/chartwerk/js/main-embed.bundle.js',
}
```

4.7.1 `CHARTWERK_EMBED_TEMPLATE`

A template string which will be rendered with context as the embed code returned to your users. The template will be rendered using the syntax of the template engine you specify in your project settings.

4.7.2 `CHARTWERK_EMBED_TEMPLATE_CONTEXT`

A function which takes one parameter, a chart instance, and returns a dictionary to use as context when rendering your template string. Any extra context you set is added to three default context items:

- `id` - the chart slug
- `size` - the preferred chart size specified by the user
- `dimensions` - stringified, escaped JSON object specifying the pixel dimensions of both chart sizes

4.7.3 Tips for configuring your embed code

While these settings give you room to completely customize your embed code, in most cases, you can easily use Chartwerk's default embed template by simply setting the `chart_path` and `embed_script` template context variables.

```
CHARTWERK_EMBED_TEMPLATE_CONTEXT = lambda chart: {
    'chart_path': 'http://www.yourawsbucket.com/path/to/your/charts/',
    'embed_script': '<CHARTWERK_DOMAIN>/chartwerk/js/main-embed.bundle.js',
}
```

Note: The `embed_script` path references the script used to inject an iframe on the parent page within your CMS. It is included with the static files in django-chartwerk, but we highly recommend you host it on S3 next to your charts.

When writing your own template string, remember that Chartwerk adds three additional pieces of context: the slug of the chart, the preferred size of the embed specified by the user and the dimensions of each chart size.

The `id` is the chart slug used to save the chart file to your S3 bucket, either `<slug>.html` or `<slug>_single.html`, for double and single-wide, respectively.

The `size` is either `double` or `single`.

The `dimensions` are a stringified JSON object specifying the height and width of both chart dimensions. You can parse it into an object and use it to set the correct dimensions of your iframe.

```
// Assuming an templated element like:
// <div data-dimensions="{{dimensions}}"></div>
var dimensions = JSON.parse(<element>.dataset.embed);

// dimensions will be an object like:
{
  double: {
    width: 500,
    height: 300,
  },
  single: {
    width: 290,
    height: 240,
  },
}
```

You can add any additional properties from your chart as template context.

Remember, that your embed template must include the scripts used to inject, configure and style the iframe on your page.

4.8 Custom templates

4.8.1 Customizing the Editor

You can customize the Editor with styles to better reflect your CMS by overriding the `chartwerk/editor.html` template. Add the template to your project and extend from `chartwerk/django-chartwerk-editor.html`.

```
<!-- chartwerk/editor.html -->
{% extends "chartwerk/django-chartwerk-editor.html" %}

{% block head_block %}
<link rel="stylesheet" type="text/css" href="some_styles.css" />
{% endblock %}

{% block body_block %}
<script src="some_script.js"></script>
{% endblock %}
```

4.8.2 Customizing the child page

If you need to customize charts' embaddable child page, you can override the template used to bake charts to S3. Add a `chartwerk/bake.html` template to your project that extends from `chartwerk/bake_base.html` and add scripts or styles within the available blocks:

```
<!-- chartwerk/bake.html -->
{% extends "chartwerk/bake_base.html" %}

{% block head_block %}
<link rel="stylesheet" type="text/css" href="some_styles.css" />
{% endblock %}

{% block body_block %}
<script src="some_script.js"></script>
{% endblock %}
```

CHAPTER 5

Using

Take a look around. Django-chartwerk consists of just a few simple views.

5.1 Start

This page lists your templates by their popularity.

Note: You can add thumbnails to your templates and add useful descriptions for when to use them in the Django admin section of your project.

5.2 Browse

This page lets you see charts created by other users in Chartwerk.

5.3 MyWerk

This page lists charts you've made, chronologically.

5.4 The Editor

The best way to learn how the editor works is to play with it. Template developers should especially check out the Code tab and the full-screen editor, which includes a tree chart of Chartwerk's internal API.

Note: This section of the docs are purposely superficial. Again, you should rely on the documentation for chartwerk-editor here.

5.5 Creating templates

You should create new templates from the ones you get for free in Chartwerk. Customize the JavaScript, CSS, HTML and dependencies to create new chart types or make existing ones adhere to your style. See chartwerk-editor's detailed template guide for complete docs.

5.6 API

You can browse django-chartwerk's API at `/api/`.

CHAPTER 6

Developing

6.1 staticapp

Front-end assets are compiled from the `staticapp` directory of django-chartwerk. This uses gulp, browserify and node-sass to compile/transpile assets to Django's standard `static` directory.

To begin developing assets, move into the `staticapp` directory and be sure to install dependencies using npm or yarn:

```
$ npm install
$ yarn
```

Then run the build process using gulp:

```
$ gulp
```

6.2 Updating chartwerk-editor

Chartwerk-editor includes a script to move compiled assets from the app into the `static` directory of django-chartwerk. First, upgrade chartwerk-editor from the `staticapp` directory.

```
$ npm update chartwerk-editor
```

Then run the script from chartwerk-editor's bin directory.

```
$ node node_modules/chartwerk-editor/bin/unbundle_django.js
```

Warning: This will overwrite chartwerk-editor templates and static files in django-chartwerk, replacing any customizations you may have made with the latest from chartwerk-editor.

6.3 Planned features

Django-chartwerk includes some models which anticipate future features.

The `TemplateProperty` and `FinderQuestion` models will be used in a future release to create an interactive wizard for picking a chart type in django-chartwerk.

CHAPTER 7

Credits

Chartwerk was developed in the newsrooms of The Dallas Morning News and POLITICO.

7.1 Developers

- Jon McClure
- Andrew Chavez

7.2 Thanks

Thanks also to these folks who've encouraged or advised development of this project in ways great and small:

- Troy Oxford
- Daniel Lathrop
- Dave Hiott
- Allan James Vestal
- Ariana Giorgi
- John Hancock
- Mike Wilson
- Ritchie King
- Gregor Aisch

CHAPTER 8

Indices and tables

- genindex
- modindex
- search