Django-Tailwind

Release 2.0.0

Tim Kamanin

Jun 19, 2022

CONTENTS

CHAPTER

ONE

DEFINITIONS

This document uses the *Tailwind* word when we talk about two things: the CSS framework and the Django package.

So let's agree that we'll use:

- **Django Tailwind**, when we talk about this very package;
- **Tailwind CSS**, when we talk about the CSS framework;

So **Django Tailwind** was created to make **Tailwind CSS** and Django play together ().

1.1 Contents

1.1.1 Installation

Step-by-step instructions

1. Install the `django-tailwind` package via `pip`:

   ```
   python -m pip install django-tailwind
   ```

   Alternatively, you can install the latest development version via:

   ```
   python -m pip install git+https://github.com/timonweb/django-tailwind.git
   ```

2. Add `'tailwind'` to `INSTALLED_APPS` in `settings.py`:

   ```
   INSTALLED_APPS = [
     # other Django apps
     'tailwind',
   ]
   ```

3. Create a *Tailwind CSS* compatible *Django* app, I like to call it `theme`:

   ```
   python manage.py tailwind init
   ```

4. Add your newly created `'theme'` app to `INSTALLED_APPS` in `settings.py`:

   ```
   INSTALLED_APPS = [
     # other Django apps
     'tailwind',
     'theme'
   ]
   ```

5. Register the generated `'theme'` app by adding the following line to `settings.py` file:

```
TAILWIND_APP_NAME = 'theme'
```

6. Make sure that the `INTERNAL_IPS` list is present in the `settings.py` file and contains the `127.0.0.1` ip address:

```
INTERNAL_IPS = [
    "127.0.0.1",
]
```

7. Install *Tailwind CSS* dependencies, by running the following command:

```
python manage.py tailwind install
```

8. The *Django Tailwind* comes with a simple `base.html` template located at `your_tailwind_app_name/templates/base.html`. You can always extend or delete it if you already have a layout.

9. If you are not using the `base.html` template that comes with *Django Tailwind*, add `{% tailwind_css %}` to the `base.html` template:

```
{% load tailwind_tags %}
...
<head>
   ...
   {% tailwind_css %}
   ...
</head>
```

   The `{% tailwind_css %}` tag includes Tailwind's stylesheet.

10. Let's also add and configure `django_browser_reload`, which takes care of automatic page and css refreshes in the development mode. Add it to `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = [
  # other Django apps
  'tailwind',
  'theme',
  'django_browser_reload'
]
```

11. Staying in `settings.py`, add the middleware:

```
MIDDLEWARE = [
  # ...
  "django_browser_reload.middleware.BrowserReloadMiddleware",
  # ...
]
```

    The middleware should be listed after any that encode the response, such as Django's `GZipMiddleware`. The middleware automatically inserts the required script tag on HTML responses before `</body>` when `DEBUG` is `True`.

12. Include `django_browser_reload` URL in your root `url.py`:

```
from django.urls import include, path
urlpatterns = [
```

(continues on next page)

(continued from previous page)

```
    ...,
    path("__reload__/", include("django_browser_reload.urls")),
]
```

13. Finally, you should be able to use *Tailwind CSS* classes in HTML. Start the development server by running the following command in your terminal:

```
python manage.py tailwind start
```

Check out the *Usage* section for information about the production mode.

Optional configurations

Content (formerly Purge) rules configuration

The `content` section of your `tailwind.config.js` file is where you configure the paths to all of your HTML templates, JavaScript components, and any other source files that contain *Tailwind* class names.

Depending on your project structure, you might need to configure the `content` rules in `tailwind.config.js`. This file is in the `static_src` folder of the theme app created by `python manage.py tailwind init {APP_NAME}`.

For example, your `theme/static_src/tailwind.config.js` file might look like this:

```
module.exports = {
    content: [
        // Templates within theme app (e.g. base.html)
        '../templates/**/*.html',
        // Templates in other apps
        '../../templates/**/*.html',
        // Ignore files in node_modules
        '!../../**/node_modules',
        // Include JavaScript files that might contain Tailwind CSS classes
        '../../**/*.js',
        // Include Python files that might contain Tailwind CSS classes
        '../../**/*.py'
    ],
    ...
}
```

Note that you may need to adjust those paths to suit your specific project layout. It is crucial to make sure that *all* HTML files (or files containing HTML content, such as `.vue` or `.jsx` files) are covered by the `content` rule.

For more information about setting `content`, check out the *"Content Configuration"* page of the Tailwind CSS docs: https://tailwindcss.com/docs/content-configuration.

Configuration of the path to the `npm` executable

Tailwind CSS requires *Node.js* to be installed on your machine. *Node.js* is a *JavaScript* runtime that allows you to run *JavaScript* code outside the browser. Most (if not all) of the current frontend tools depend on *Node.js*.

If you don't have *Node.js* installed on your machine, please follow installation instructions from the official Node.js page.

Sometimes (especially on *Windows*), the *Python* executable cannot find the `npm` binary installed on your system. In this case, you need to set the path to the `npm` executable in *settings.py* file manually (*Linux/Mac*):

```
NPM_BIN_PATH = '/usr/local/bin/npm'
```

On *Windows* it might look like this:

```
NPM_BIN_PATH = r"C:\Program Files\nodejs\npm.cmd"
```

Please note that the path to the `npm` executable may be different for your system. To get the `npm` path, try running the command `which npm` in your terminal.

1.1.2 Usage

Running in development mode

To start Django Tailwind in development mode, run the following command in a terminal:

```
python manage.py tailwind start
```

This will start a long-running process that watches files for changes. Use a combination of `CTRL + C` to terminate the process.

Several things are happening behind the scenes at that moment:

1. The stylesheet is updated every time you add or remove a CSS class in a Django template.
2. The `django-browser-reload` watches for changes in HTML and CSS files. When a Django template or CSS is updated, browser refreshes them. That gives you a smooth development experience without the need to reload the page to see updates.

Building for production

To create a production build of your theme, run:

```
python manage.py tailwind build
```

This will replace the development build with a bundle optimized for production. No further actions are necessary; you can deploy!

1.1.3 Settings

Django Tailwind comes with preconfigured settings. You can override them in the `settings.py` of your Django project.

TAILWIND_APP_NAME

This defines the *Tailwind* theme Django app containing your *Tailwind CSS* styles. I prefer to name such an app `'theme'`. You should generate the app during the installation phase by running the following command:

```
python manage.py tailwind init
```

Please refer to the *Installation* section for more information on the installation process.

TAILWIND_DEV_MODE (in deprecation)

Determines whether the `browser-sync` snippet is added to the page via the `{% tailwind_css %}` tag. It is set to `False` by default, so if you use legacy pre-`3.1.0` configuration and rely on `browser-sync`, add `TAILWIND_DEV_MODE=True` to your `settings.py`.

NPM_BIN_PATH

This defines a path to the `npm` executable in your system.

> *Tailwind CSS* requires you to have *Node.js* installed on your machine. *Node.js* is a *JavaScript* runtime that allows running *JavaScript* code outside a browser. Most of the current frontend tools depend on *Node.js*.
>
> If you don't have *Node.js* installed on your machine, please follow installation instructions from the official Node.js page.

The default value is:

```
NPM_BIN_PATH = 'npm'
```

Please note, that on *Windows* the path might look completely different (pay attention to "backslashes" in the path):

```
NPM_BIN_PATH = r"C:\Program Files\nodejs\npm.cmd"
```

TAILWIND_CSS_PATH

This defines a path to the generated *Tailwind CSS* stylesheet. If you have created a theme app via the `python manage.py tailwind init` command, chances are you don't need to change this value.

However, if you integrated *Tailwind CSS* in another way or want to use a *CDN* version of the bundle, you might want to change the path.

The default value is:

```
TAILWIND_CSS_PATH = 'css/dist/styles.css'
```

1.1.4 Template tags

Django Tailwind introduces a couple of template tags for your convenience.

{% tailwind_css %} tag

Usage

The `{% tailwind_css %}` tag generates a stylesheet link for the `'theme'` app and that's all you need to include *Tailwind's CSS* on a page:

```
{% load tailwind_tags %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Django Tailwind</title>
    {% tailwind_css %}
  </head>
  <body></body>
</html>
```

Asset versioning

The tag also supports asset versioning via the `v=` parameter, like this:

```
{% tailwind_css v='1' %}
```

Depending on your production setup, you may or may not need this functionality, so it's optional.

{% tailwind_preload_css %} tag

The tag generates a preload directive for your stylesheet, which improves loading performance in production. Place it above the `{% tailwind_css %}` tag:

```
{% load tailwind_tags %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Django Tailwind</title>
    {% tailwind_preload_css %}
    {% tailwind_css %}
  </head>
  <body></body>
</html>
```

It also supports asset versioning (if needed):

```
{% tailwind_preload_css v='1' %}
```

1.1.5 Migrating from browser-sync to django-browser-reload

Starting with version 3.1.0, *Django-Tailwind* is dropping support for *Node.JS*-based browser-sync and switching to *Django*-based django-browser-reload as the primary solution for seamless page and asset updates during development.

If you generated *Tailwind CSS* before 3.1.0, I strongly suggest you to upgrade your project.

> If you don't want to upgrade and are happy with browser-sync, make sure that you have TAILWIND_DEV_MODE=DEBUG set in your settings.py file.

Please follow the following migration steps.

1. Upgrade *Django-Tailwind* to its latest version:

```
pip install django-tailwind --upgrade
```

2. Add django_browser_reload to INSTALLED_APPS in settings.py:

```
INSTALLED_APPS = [
  # other Django apps
  'tailwind',
  'theme',
  'django_browser_reload'
]
```

3. Staying in settings.py, add the middleware:

```
MIDDLEWARE = [
  # ...
  "django_browser_reload.middleware.BrowserReloadMiddleware",
  # ...
]
```

   The middleware should be listed after any that encode the response, such as Django's GZipMiddleware. The middleware automatically inserts the required script tag on HTML responses before </body> when DEBUG is True.

4. Include django_browser_reload URL in your root url.py:

```
from django.urls import include, path
urlpatterns = [
    ...,
    path("__reload__/", include("django_browser_reload.urls")),
]
```

5. Now, cd to your TAILWIND_APP_NAME static_src directory. In my case, TAILWIND_APP_NAME is theme, thus I cd to theme/static_src:

```
cd theme/static_src
```

6. Remove bs.config.js file from the theme/static_src directory.

7. Open package.json file in theme/static_src directory, and remove the following commands under scripts: sync, dev:sync, dev.

8. Staying in package.json, rename the dev:tailwind command under scripts to dev. That's the only command we need for development from now on.

9. Staying in the same directory, uninstall dependencies we don't need anymore, run the following command:

```
npm uninstall browser-sync nodemon npm-run-all
```

10. Run `python manage.py tailwind start` to make sure everything runs properly.

1.1.6 Upgrading from v2 to v3

If you're coming from *Django-Tailwind* `2.x`, your *Tailwind CSS* project probably depends on *Tailwind CSS* `2.x`. If you want to use latest features of the version `3.x`, you need to upgrade your *Tailwind CSS* app from `2.x` to `3`.

In the following instructions, I assume that your *Tailwind CSS* app name is set as `TAILWIND_APP_NAME = 'theme'` in `settings.py`.

If it's different for you, please replace the `theme` with an app name of your choice, while following my steps.

Let's do this!

1. Update *Django-Tailwind* to `3.x` by running the following command:

```
pip install --upgrade django-tailwind
```

2. Next, update dependencies and plugins via `npm`. Go to **theme/static_src** directory and run the following command:

```
npm install -D tailwindcss@latest \
  @tailwindcss/typography@latest \
  @tailwindcss/forms@latest \
  @tailwindcss/aspect-ratio@latest \
  @tailwindcss/line-clamp@latest \
  postcss@latest \
  autoprefixer@latest
```

3. Now, open **theme/static_src/bs.config.js** and replace `...tailwindConfig.purge` with `...tailwindConfig.content`,
4. Then, open **theme/static_src/tailwind.config.js** in a code editor, and
5. Remove the `mode` property. In Tailwind `3.0`, `jit` is always on, so there's no use of `mode: jit` string anymore,
6. Rename `purge` to `content`,
7. Remove dark mode configuration by deleting the line containing `darkMode`,
8. Staying with the open **theme/static_src/tailwind.config.js** file, refer to the official Tailwind CSS upgrade guide and see if there's anything else you need to change in your **tailwind.config.js**. In most cases everything should already be fine, but depending on your configuration, some parts may need your attention.

And that's it. You're now on the latest Django-Tailwind `3.0` with the latest Tailwind CSS `3.0` installed and ready to go.

1.1.7 Updating *Tailwind CSS* and its dependencies

When a new release of *Tailwind CSS* comes out, you can update your `theme` project without updating *Django Tailwind*.

Checking if there are updates for *Tailwind CSS* and its dependencies

Before doing an update, you can check if there are any updates. Run the following command:

```
python manage.py tailwind check-updates
```

This command runs the `npm outdated` command behind the scenes in the context of your `theme/static_src` directory.

If there are updates, you'll see a table of dependencies with the latest compatible versions. If there are no updates, this command will return no output.

Updating Tailwind CSS and its dependencies

If you want to use the latest version of *Tailwind CSS*, run the following command:

```
python manage.py tailwind update
```

This command runs the `npm update` command behind the scenes in the context of your `theme/static_src` directory.

If there are updates, you'll see a log of updated dependencies. If there are no updates, this command will return no output.

1.1.8 Running in Docker

You can find a Docker example under the `example` directory.

Check the included `example/Dockerfile` and `example/docker-compose.yml` for more information.

Here's how to start the `example` project via Docker:

1. Go into the `example` directory;

   ```
   cd example
   ```

2. Build containers via `docker-compose`:

   ```
   docker-compose build
   ```

3. Start containers:

   ```
   docker-compose up
   ```

4. Open `http://localhost:8000` in a browser. You should see the main page.