django-auto-webassets Documentation

Release 0.0.1

Thomas Hartmann

Jan 02, 2020

Contents:

1 Reference **3**
- 1.1 Settings . . . 3
- 1.2 Template Tags . . . 3
- 1.3 Mixins . . . 4
- 1.4 Commands . . . 4

2 Quickstart **5**
- 2.1 Prerequisites . . . 5
- 2.2 Install . . . 5
- 2.3 Configure . . . 6
- 2.4 The use case . . . 6
- 2.5 Setting up the base template . . . 7
- 2.6 Using a different javascript file in a view . . . 7
- 2.7 Build assets by command . . . 7

3 Indices and tables **9**

Python Module Index **11**

Index **13**

django-auto-webassets is a Django app that automates the creation of webassets/django-assets bundles. Besides these python libraries, it makes heavy use of requirejs.

The library was written for one particular use-case in mind which will be described here. Contributions are always welcome!

CHAPTER 1

Reference

1.1 Settings

django-auto-webassets provides the following Django settings:

WEBASSETS_JS_FOLDER Standard folder for the javascript files of the views. Default: `js_for_views`

WEBASSETS_PATH_TO_REQUIREJS The full relative path to `requirejs.js` relative to one place that Django's staticfile finder algorithm searches. Default: `requirejs/require.js`

WEBASSETS_PATH_TO_REQUIREJS_CFG The full relative path to `requirejs.cfg.js` relative to one place that Django's staticfile finder algorithm searches. Default: `requirejs_cfg.js`

WEBASSETS_SCRAMBLE_OUT_JS By default, the name of the generated javascript file contains the full module path of the view. This should not lead to any security risks because the code of the web application should not be accessible anyhow to the user. Anyhow, if you want uuid names, set this to true. Default: `False`.

WEBASSETS_R_JS The full relative path to `r.js` relative to one place that Django's staticfile finder algorithm searches. Default: `node_modules/.bin/r.js`

WEBASSETS_OPTIMIZE If this is set to true, standard optimizations (minifying etc.) are run on the resulting javascript file. Default: `True`.

1.2 Template Tags

django-auto-webassets provides one template tag to include the generated bundle in a template.

webassets_js You should put this tag somewhere in your template where javascript is expected. Please refer to the *main documentation*.

1.3 Mixins

class django_auto_webassets.views.mixins.**AutoWebassetsJSMixin**(**args*, ***kwargs*)
View Mixin to specify view-specific javascript bundle.

full_webasset_path
Return the full path to the javascript file.

classmethod get_webasset_bundle_name()
Return the name of the webasset bundle.

classmethod get_webasset_js_file()
Return the filename of the javascript file.

1.4 Commands

webassetsrequirejs Build assets defined by the *django_auto_webassets.views.mixins.AutoWebassetsJSMixin* mixin.

CHAPTER 2

Quickstart

2.1 Prerequisites

I assume that you have nodejs including npm installed. I also assume that the required javascript packages are declared in the `package.json` file and `npm update` installs these in a folder called `node_modules`. Most importantly, requirejs must be one of the required javascript packages.

I also assume that one `.js` file exists that configures requirejs. Here is an example:

```
require.config({
    "baseUrl": "/static",

    "nodeRequire": "require",

    "shim": {
        "bootstrap": {"deps": ["jquery"]},
    },

    "paths": {
        "jquery": "jquery/dist/jquery",
        "bootstrap": "bootstrap-sass/assets/javascripts/bootstrap",
    },
});
```

2.2 Install

```
pip install django-auto-webassets
```

This will also install django-assets and all necessary dependencies.

2.3 Configure

Next, include django-assets and django-auto-webassets in the `INSTALLED_APPS` section of the Django `settings.py`:

```
INSTALLED_APPS = [
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django_assets',
    'django_auto_webassets'
    ]
```

And also make sure to include `django_assets.finders.AssetsFinder` in `STATICFILES_FINDERS`:

```
STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
    'django_assets.finders.AssetsFinder',
)
```

Next, make sure that the `node_modules` folder is included in the `STATICFILES_DIRS`. If the `node_modules` folder is in the base folder of your project, this would look like this:

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'node_modules'),
    ...
]
```

Also, make sure to the django-auto-webassets where it finds the configuration file for requirejs:

```
WEBASSETS_PATH_TO_REQUIREJS_CFG = 'requirejs_cfg.js'
```

For now, I recommend that you turn off optimization and turn on assets debugging:

```
WEBASSETS_OPTIMIZE = False
ASSETS_DEBUG = False
```

2.4 The use case

The use case for this library is that your templates and views are hierarchical and reusable. You basically have one `base.html` template that basically defines your theme. Then you extend this template to create one template that displays lists and maybe another one that displays and processes forms.

As each child-template becomes more specialized, so does the javascript. There is some javascript that you need on any given page. A page that displays a list might need the general purpose javascript plus some list-displaying specifics. A specific view might then need to add the specific javascript for that view but might be able to use the generic list-displaying template.

django-auto-webassets in conjunction with requirejs makes it really easy to do this adhering to the DRY principle.

2.5 Setting up the base template

Assuming you have a javascript file called `init_general.js` that does the general javascript stuff for your pages, add this to your base template:

```
{% load auto_webassets %}

{% block javascript %}
   {% webassets_js "init_general.js" %}
{% endblock %}
```

The `webassets` tag automatically adds requirejs, the requirejs configuration file you defined above and `init_general.js` to the html output.

Depending on the configuration, it also applies optimization (e.g., bundling...) and adds the resulting file to the html output.

If you want to use a different javascript file in a child-template, just override the `javascript` block.

2.6 Using a different javascript file in a view

You can use *`django_auto_webassets.views.mixins.AutoWebassetsJSMixin`* to specify a different javascript file for that specific view.

Attention: Always include mixins **before** the view class!

```
from django_auto_webassets.views.mixins import AutoWebassetsJSMixin
from django.views.generic import TemplateView

class MyView(AutoWebassetsJSMixin, TemplateView):
   template_name = 'generic.html'
   webasset_js_file = 'specific_javascript.js'
```

Now `specific_javascript.js` is used instead of the javascript file specified in the template.

2.7 Build assets by command

When optimization is turned on by setting `WEBASSETS_OPTIMIZE = True` in the `settings.py`, webassets needs to build and optimize the javascript for the website. It normally does that when the specific page is requested for the first time. However, this takes a few seconds which is not good for the user experience.

Instead, you can use the management command that comes with django-auto-webassets to pre-build all bundles:

```
python manage.py webassetsrequirejs
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

django_auto_webassets.views.mixins, 4

Index

A

AutoWebassetsJSMixin (*class in django_auto_webassets.views.mixins*), 4

D

django_auto_webassets.views.mixins (*module*), 4

F

full_webasset_path (*django_auto_webassets.views.mixins.AutoWebassetsJSMixin attribute*), 4

G

get_webasset_bundle_name() (*django_auto_webassets.views.mixins.AutoWebassetsJSMixin class method*), 4

get_webasset_js_file() (*django_auto_webassets.views.mixins.AutoWebassetsJSMixin class method*), 4