Django Assess Managed Nicely Documentation

Release 0.2.0

Curtis Maloney

August 06, 2014

Contents

CHAPTER 1

Settings

1.1 DAMN_PROCESSORS

Default: {}

A map of Processor configs.

Each value is a dict of config values. The only required option is 'processor', which is an import path for the class to use to process this asset type.

```
DAMN_PROCESSORS = {
    'js': {
        'processor': 'damn.processors.ScriptProcessor',
        'aliases': {
            'jquery': 'js/vendor/jquery-1.11.min.js',
            'jqplot': 'js/vendor/jqploy-0.9.min.js',
        },
        'deps': {
            'jqplot': ['jquery',],
        },
    },
}
```

See *Processors* for more details.

1.2 DAMN_MODE_MAP

Default: {}

A map of file extensions to mode names.

In the absense of an entry here, or an explicit `mode=` attribute on the asset tag, the asset will be assigned to a mode with the same name as its file extension.

1.3 DAMN_MODE_ORDER

Default: ['css', 'js',]

A list of the order in which asset types should be rendered.

CHAPTER 2

Template Tags

2.1 Assets tag

This tag simply marks where to output the asset tags.

```
{% assets %}
```

2.2 Asset tag

Specifies an asset that is required for this page to function.

```
{% asset name ...deps... [mode=?] [alias=?] %}
```

The first positional argument is the filename or alias of an asset that is required.

Additional positional arguments are names or aliases of assets this asset requires to be included before it.

The `mode` keyword allows you to override which Processor this asset will be assigned to. By default it will be assined to the Processor with the name matching the assets file extension, mapped through `DAMN_MODE_MAP`

CHAPTER 3

Processors

The processor takes the list of assets and renders the output to the page.

As assets are defined in the page, the AssetRegister will assign them to Processors, which it will in turn request to render them.

class `Processor`

`aliases`
A map (dict) of aliases to filenames.

This is useful for removing the need to, for instance, remember which version of jQuery your site is using.

`deps`
A map of filenames or aliases to lists of other assets they depend on.

Any use of these assets will automatically add the listed dependencies.

`add_asset`(*filename*, *alias*, *deps*)
Add an asset to this `Processor`. This is used by the `{% asset %}` tag.

`resolve_deps`()
Return a list of filenames in an order which respects the declared dependencies. All aliasese will be resolved at this point.

`alias_map`(*name*)
Unalias a given resource name.

Out of the box there are two processors: ScriptProcessor, and LinkProcessor.

3.1 ScriptProcessor

`ScriptProcessor` will output each asset in a script tag, after resolving the filename through staticfiles.

```
<script src="{% static filename %}"></script>
```

3.2 LinkProcessor

`LinkProcessor` will output each asset as a link tag. You can optionally specify in the config the `rel` and `type` attributes to be used.

```
<link rel="{{ rel }}" type="{{ type }}" href="{% static filename %}">
```

class LinkProcessor

rel
Default: 'stylesheet'

The value to output for the rel attribute of the link element.

type
Default: 'text/css'

The value to output for the type attribute of the link element.

CHAPTER 4

Changelog

4.1 v0.2.1

4.1.1 Bugs Fixed

- Fix forgotten var rename in exception

4.2 v0.2.0

New Features

- Added more tests
- Deps can now be new aliases
- Aliases are implicitly added to Deps
- Catch dependencies that appear to be missing aliases

4.3 v0.1.1

Initial Release

CHAPTER 5

Overview

Yet another approach to handling static assets in your Django project.

Django-AMN helps you manage including assets in your templates without requiring contortions to enable adding extra dependencies, only to have duplicates appearing.

Simply add the *{% assets %}* tag where you want your assets listed, and then use the *{% asset %}* tag to add a requirement.

Assets can have dependencies on other assets, so you can be sure they're included, and in the right order.

Different asset types [css, js, less, etc] will be assigned to a different *Processor*, which can handle how they're rendered into the template - including compiling, translating, minifying, etc.

Each asset is assigned a mode (by default its file extension) which you can override by specifying it in the tag:

```
{% asset 'thing/foo.html' mode='template' %}
```

Assets can have dependencies, so you won't forget to include what's needed.

```
{% asset 'js/knockout.js' 'js/jquery.js' %}
```

You can also pre-define asset dependencies in your *settings*.

To make life easier, any asset can have an alias. Aliases can be assigned in two ways; either in the tag, or in your settings.

```
{% asset 'js/jquery-1.11.min.js' alias='jquery' %}
```

Dependencies can refer to aliases, allowing library versions to be updated without breaking your templates.

```
{% asset 'js/knockout.js' 'jquery' %}
```

If you've configured aliases in your settings, you can use them directly:

```
{% asset 'jqplot' %}
```

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*