django-require-license Documentation

Release 1.0.1

Collab

January 08, 2016

Contents

1 Installation **3**

2 Tutorial **5**
2.1 Tutorial 5
2.2 Development 8

Django staticfiles post-processor that allows you to prepend a license header to the compressed file(s) generated by django-require.

CHAPTER 1

Installation

Use pip to download and install the package from PyPi:

```
pip install django-require-license
```

Or checkout the source code from Github:

```
git clone https://github.com/collab-project/django-require-license.git
```

CHAPTER 2

Tutorial

Read the Tutorial for more information on how to configure your django-require and Require.js project.

2.1 Tutorial

This tutorial shows how to prepend a license header to one or more compressed files in your django-require project.

2.1.1 Setup Require.js Application

Start by creating and configuring the require.js application. If your application is already up and running, you can skip to the *Django configuration* section.

Structure

For this example, in a Django project called `myproject`, there is a require.js application called `app` with a directory structure similar to this:

```
+ myproject
  - manage.py
  + myproject
  + static
    + js
      + libs
        - almond.js
        - jquery.min.js
        - loglevel.js
        - require.js
      + app
        - main.js
      - app.build.js
      - app.js
      - JS-LICENSE.txt
```

Main Application

The main require.js application in `static/js/app/main.js` contains:

```
define([
  'jquery',
  'loglevel'
], function($, log)
{

  log.enableAll();
  log.debug("Hello world!");

});
```

So we import the loglevel library as `log`, enable logging on all levels and print a test message.

Build Config

The r.js build configuration file `static/js/app.build.js` contains:

```
({
    mainConfigFile: "app.js",
    name: "app",
    preserveLicenseComments: false,
    skipDirOptimize: true
})
```

Check the r.js docs for an explanation of these options.

Application Config

The application configuration file `static/js/app.js` contains:

```
require.config({
    paths: {
        jquery:         'libs/jquery.min',
        almond:         'libs/almond',
        loglevel:       'libs/loglevel'
    },
    shim: {
        loglevel: {
            exports:    'loglevel'
        },

        almond: {
            exports:    'almond'
        }
    }
});

// Load the main app module to start the app
require(['app/main']);
```

Libraries

The libraries used in this example are placed in `static/js/libs`.

License File

`static/js/JS-LICENSE.txt` is a plain-text file containing the license text. You can use variable placeholders that are replaced during the build:

```
/*! Copyright {copyright_holder} {copyright_year} - v{version} ({timestamp})
 *  {license_url}
*/
```

2.1.2 Configure Django application

Make sure the staticfiles and django-require applications are included in the `INSTALLED_APPS` setting of the Django project settings file:

```
INSTALLED_APPS = [
  # ...
  'django.contrib.staticfiles',
  'require'
]
```

The `STATIC_ROOT` setting points to an absolute directory path where the static files should be collected to:

```
STATIC_ROOT = '/path/to/static/'
```

Change the `STATICFILES_STORAGE` setting to `require_license.storage.OptimizedStaticFilesStorage`:

```
# The file storage engine to use when collecting static files with the
# `collectstatic` management command.
STATICFILES_STORAGE = 'require_license.storage.OptimizedStaticFilesStorage'
```

Configure the django-require application (refer to the documentation for details):

```
import os

# The baseUrl to pass to the r.js optimizer.
REQUIRE_BASE_URL = 'js'

# The name of the build profile for the site, relative to REQUIRE_BASE_URL.
# Leave blank to use the built-in default build profile.
REQUIRE_BUILD_PROFILE = 'app.build.js'

# The name of the require.js script used by your project, relative to
# REQUIRE_BASE_URL.
REQUIRE_JS = os.path.join(REQUIRE_BASE_URL, 'libs', 'require.js')

# Whether to run django-require in debug mode.
REQUIRE_DEBUG = DEBUG

# A dictionary of standalone modules to build with almond.js.
REQUIRE_STANDALONE_MODULES = {
    'app': {
        # Where to output the built module, relative to REQUIRE_BASE_URL.
        'out': 'app.min.js',

        # A build profile used to build this standalone module.
        'build_profile': REQUIRE_BUILD_PROFILE,
    }
}
```

```
# A tuple of files to exclude from the compilation result of r.js.
REQUIRE_EXCLUDE = (
    'build.txt',
    os.path.join(REQUIRE_BASE_URL, REQUIRE_BUILD_PROFILE),
)
```

Configure Header

Configure the `REQUIRE_LICENSE_HEADERS` options. This is a dict object where you add a mapping for the output file (eg. `js/app.min.js`) and a dict containing the variables that we inject into the license header.

```
from datetime import datetime, date

# A dictionary of output files with a license header config.
REQUIRE_LICENSE_HEADERS = {
    os.path.join(REQUIRE_BASE_URL, 'app.min.js'): {
        'license_file': os.path.join(REQUIRE_BASE_URL, 'JS-LICENSE.txt'),
        'timestamp': date.today(),
        'copyright_year': datetime.now().year,
        'copyright_holder': 'MyCompany',
        'license_url': 'http://example.com/license',
        'version': 'myproject.version'
    }
}
```

The only mandatory key is `license_file`: the path to the license header template file, eg. `js/JS-LICENSE.txt`.

The `version` key is special: use a string value here, eg. `1.0.4`, or specify a fully-qualified path to an attribute that contains a string version instead, eg. `myproject.version`.

Any other keys found in the dict will also be injected in the license header template.

2.1.3 Optimize

Now you're ready to run the `collectstatic` command to collect and optimize the static files:

```
./manage.py collectstatic
```

This copies all static files into the `STATIC_ROOT` directory, including the compressed `app.min.js` with license header.

2.2 Development

After checkout, install package in active virtualenv:

```
pip install -e .
```

2.2.1 Testing

Running tests with Tox:

```
tox -v
```

Running tests without Tox:

```
./runtests.py
```

Directly with `django-admin`:

```
django-admin test --settings=require_license.tests.settings require_license
```

2.2.2 Coverage

To generate a test coverage report using coverage.py:

```
coverage run --source='.' runtests.py
coverage html
```

The resulting HTML report can be found in the `htmlcov` directory.