Django-Systemjs Documentation

Release 1.4.1

Sergei Maertens

December 14, 2016

Contents

Modern Javascript in Django.

CHAPTER 1

Overview

Django SystemJS brings the Javascript of tomorrow to Django, today.

It leverages JSPM (https://jspm.io) to do the heavy lifting for your client side code, while keeping development flow easy and deployment without worries. In DEBUG mode, your Javascript modules are loaded asynchronously. In production, your app is nicely bundled via JSPM and ties in perfectly with `django.contrib.staticfiles`.

CHAPTER 2

Requirements

Django-SystemJS runs on Python 2.7, 3.3, 3.4 and 3.5. Django versions 1.8, 1.9 and 1.10 are supported.

CHAPTER 3

Contents

3.1 Installation

3.1.1 Django

Install by running:

```
pip install django-systemjs
```

Then, add `systemjs` to your `settings.INSTALLED_APPS`, and add the custom staticfiles finder:

```
STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
    'systemjs.finders.SystemFinder',
]
```

The custom finder looks up files in `STATIC_ROOT` directly. It is not needed if you use `django-compressor` and have `COMPRESS_ROOT` set to `STATIC_ROOT` (which is the default).

If you want to use `django.contrib.staticfiles.storage.ManifestStaticFilesStorage` as staticfiles storage backend, you need to use the systemjs-version: `systemjs.storage.SystemJSManifestStaticFilesStorag`
This storage ensures two things:

- during bundling the collected staticfiles (from `collectstatic`) aren't removed from the manifest file.
- during collectstatic the bundled files are kept in the manifest.

There are some *application settings* to customize behaviour. Usually you shouldn't need to change these, since they have sane defaults.

3.1.2 JSPM

Since you found this django app, you probably already know what jspm is and how to install it. There is some django-specific configuration to be done to make everything play nicely together.

Note: Currently there are two jspm branches active: 0.16.x and 0.17.x (beta). The beta has some backwards incompatibilities with 0.16, and most of all, some great features to speed up development. In general, I recommend using 0.17.x, even more so if you're going to do React.js stuff.

Setting up jspm for the first time

If you have worked with jspm before, you can skip this.

To install `jspm` (http://jspm.io), you'll need some front-end tooling. Make sure you have `nodejs` and `npm` installed on your system. See the nodejs installation instructions.

If you never installed `jspm` before, install it globally for the first time:

```
sudo npm install -g jspm
```

This ensures that the `jspm` cli is available in your `$PATH`.

jspm 0.17.x (beta) instructions

There is an example project with detailed installation and configuration information, aimed at the 0.17 beta version of jspm. The README.md contains a step-by-step history to get to a working setup with the standard Django project layout.

jspm 0.16.x (stable) instructions

jspm uses the `package.json` from NodeJS, so get that set-up first:

```
npm init
```

This will bring up an interactive prompt to ask for some package information.

Next, install `jspm` locally in your project, and pin its version:

```
npm install --save-dev jspm
```

It's now time to initialize your `jspm` project. This is an interactive prompt again, but we'll need to deviate from the defaults a bit to make it play nice with Django.

```
jspm init

Would you like jspm to prefix the jspm package.json properties under jspm? [yes]: yes  # easier to ke

Enter server baseURL (public folder path) [/]: static  # same as settings.STATIC_ROOT, relative to pa

Enter jspm packages folder [static/jspm_packages]:  # keep it within settings.STATIC_ROOT

Enter config file path [static/config.js]: my-project/static/config.js  # must be kept in version co

Enter client baseURL (public folder URL) [/]: /static/ # set to settings.STATIC_URL

Do you wish to use a transpiler? [yes]: # current browsers don't have full support for ES6 yet

Which ES6 transpiler would you like to use, Traceur or Babel? [traceur]: babel  # better tracebacks
```

Take some time to read the jspm docs if you're not familiar with it yet.

Note: A few settings are remarkable. We put `jspm_packages` in `settings.STATIC_ROOT`. This means that collectstatic will not post-process the files in here, which can be a problem. Django-SystemJS deals with this specific use case as it is intended for `jspm`-users. There is an inherent limitation within jspm which should be lifted with the 0.18 release.

3.2 Usage

3.2.1 Template tag

Usually, in your template you would write something like:

```
<script src="/static/jspm_packages/system.js"></script>
<script src="/static/config.js"></script>
<script>System.import('my/awesome/app.js');</script>
```

With Django SystemJS you can replace this with:

```
{% load staticfiles system_tags %}

<script src="{% static 'jspm_packages/system.js' %}"></script>
<script src="{% static 'config.js' %}"></script>
{% systemjs_import 'my/awesome/app.js' %}
```

In development mode (`DEBUG = True`), the tag will output the previous `System.import('my/awesome/app.js');` statement. In production mode, it will output:

```
<script src="/static/SYSTEMJS/my/awesome/app.js"></script>
```

This url is generated by the configured static files backend, so if you use the `CachedStaticFilesStorage`, you will get the hashed path:

```
<script src="/static/SYSTEMJS/my/awesome/app.12ab459edf22.js"></script>
```

Note: django-storages(-redux) and S3 is untested. If you run into any issues, please raise an issue on Github.

If you want to use `django.contrib.staticfiles.storage.ManifestStaticFilesStorage`, you need to use the systemjs-version: `systemjs.storage.SystemJSManifestStaticFilesStorage`. This storage ensures that during bundling the collected staticfiles (from `collectstatic`) aren't removed from the manifest file.

The tag accepts any number of (custom) attributes you may want to add, making it possible to load scripts async, or specify html ids for example.

```
{% load system_tags %}
{% systemjs_import 'my/awesome/app.js' async id="my-awesome-app" %}
```

this will output (in production mode)

```
<script type="text/javascript" async id="my-awesome-app" src="/static/SYSTEMJS/my/awesome/app.12ab459
```

3.2.2 Management commands

Django-SystemJS comes with a few management commands to create and manage all the bundles. It does so by checking all your template files and extracting the `{% systemjs_import '...' %}` nodes.

systemjs_bundle

Process the project templates, extract the apps and bundle them with jspm.

```
python manage.py systemjs_bundle
```

By default it will look at all templates in your app directories, and the additional template dirs for the vanilla Django template engine - Jinja2 is unsupported.

Supporting all the jspm command line arguments is work in progress. Currently the following options are supported:

Options

- `--minify`: passes the `--minify` flag down to jspm to generate minified bundles
- `--sfx`: generates a self-executing bundle.
- `--template, -t`: pass the name of a template (for example `myapp/base.html`), and Django SystemJS will only look in those files for imported apps. It will no longer parse all project templates. This option can be specified multiple times to look in a set of templates.
- `--minimal`: the minimal option will only rebundle apps that have changed. If you have multiple `{% systemjs_import <...> %}` statements, and only one app was changed, this can speed up the total bundle time. Comparison happens based on mtimes and md5 hashes of the involved files.

  Note: Changes to the source files for the bundles are detected, but changes to jspm config files (`jspm.config.js`, `jspm.browser.js`) are not included. These files can change relatively frequently, hereby invalidating the depcache when it's not needed. Be careful when making bundle-altering config file changes.

  The first time you use the `--minimal` option, you will get an error saying that the `deps.json` file cannot be located. This is because we have never written the dependency tree yet.

  You can do this manually the first time by executing the *systemjs_write_depcaches* command.
- `--node-path`: path to the `node_modules` directory of your project. Required if Django-SystemJS cannot figure it out by itself and the `NODE_PATH` environment variable is not set.

systemjs_show_packages

Parses the templates and reports the apps found in them. Useful to get a quick overview of all the bundles to be generated.

systemjs_write_depcaches

Parses the templates and extracts the apps found in them. For every app, the dependencies are traced and written to disk. This depcache is used with the `--minimal` option of the *systemjs_bundle* command.

Note: If you bundle with any of the `--sfx`, `--minimal` or `minify` options, you need to use the same options to write the depcache. A difference in bundle options will trigger a re-bundle.

3.2.3 Example workflow

Django SystemJS is designed as a non-intrusive library in development mode, so that it won't sit in your way too much. Simply using the template tag will be all you have to do as long as you're running with `DEBUG=True`.

Example steps for deployment:

- Run `git pull` to update your copy of the code
- Install the dependencies: `npm install`, followed by `jspm install`
- Run collectstatic: `python manage.py collectstatic`
- Bundle the apps in your project: `python manage.py systemjs_bundle`.

The order of operations matters: to bundle, all the bits and pieces must be collected so that `jspm` can retrieve them in your `STATIC_ROOT`. It has no notion of your `static` folders within your apps.

3.3 Available settings

`SYSTEMJS_ENABLED`: defaults to `not settings.DEBUG`. If disabled, the loading of modules will happen in the 'standard' jspm way (transpiling in browser).

`SYSTEMJS_JSPM_EXECUTABLE`: path to the `jspm-cli` executable. Defaults to `jspm`, which should be available if installed globally with `npm`.

`SYSTEMJS_OUTPUT_DIR`: name of the subdirectory within `settings.STATIC_ROOT`. Bundled files will end up in this directory, and this is the place the templatetag will point static files to.

`SYSTEMJS_PACKAGE_JSON_DIR`: directory containing your `package.json` file. This is automatically guessed from `BASE_DIR`. You will get an error in the shell if you need to set it yourself.

`SYSTEMJS_DEFAULT_JS_EXTENSIONS`: in prior verions of jspm, the `.js` extension for imports was optional. This is being phased out, and matches the `defaultJSExtensions` settings in `config.js`.

`SYSTEMJS_CACHE_DIR`: directory to keep the dependency cache (when generating *minimal* bundles)

`SYSTEMJS_SERVER_URL`: if you're using a frontend asset-server and want to use that instead of letting Django serve the modules, specify the url with this settings. Defaults to `None`. Example: `http://localhost:3000/assets/`.

3.3.1 Environment variables

`NODE_PATH`

When generating *minimal* bundles, a NodeJS script (`trace-deps.js`) is called. This script needs to be called from the directory containing `package.json`. If Django-SystemJS cannot figure out this directory by itself, you may need to set the environment variable:

```
export NODE_PATH=/path/to/project/node_modules
```

3.4 Contributing

To get up and running quickly, fork the github repository and make all your changes in your local clone.

Git-flow is prefered as git workflow, but as long as you make pull requests against the `develop` branch, all should be well. Pull requests should always have tests, and if relevant, documentation updates.

Feel free to create unfinished pull-requests to get the tests to build and get work going, someone else might always want to pick up the tests and/or documentation.

Add yourself to the CONTRIBUTORS file if you want, we want you to be proud of your work!

3.4.1 Testing

Django's testcases are used to run the tests.

To run the tests in your (virtual) environment, simple execute

```
python runtests.py
```

This will run the tests with the current python version and Django version installed in your virtual environment.

To run the tests on all supported python/Django versions, use tox.

```
pip install tox
tox
```

If you want to speed this up, you can also use detox. This library will run as much in parallel as possible.

3.4.2 Documentation

The documentation is built with Sphinx. Run *make* to build the documentation:

You can now open *_build/index.html*.

3.4.3 Coding style

Please stick to PEP8, and use pylint or similar tools to check the code style. Also sort your imports, you may use `isort` for this. In general, we adhere to Django's coding style.

CHAPTER 4

License

Licensed under the MIT License.

CHAPTER 5

Souce Code and contributing

The source code can be found on github.

Bugs can also be reported on the github repository, and pull requests are welcome. See *Contributing* for more details.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search