django-easy-pjax Documentation

Release 1.3.0

Filip Wasilewski

February 21, 2016

Contents

Enhance the browsing experience of Django sites. Developed at en.ig.ma software shop.

CHAPTER 1

What is PJAX?

PJAX utilizes pushState and Ajax to load HTML from the server into the current page without a full reload. It's Ajax with real permalinks, page titles, and a working back button that fully degrades.

Check out the demo that illustrates this concept in practice and take a look at docs of jquery-pjax to get more information.

The `django-easy-pjax` app is a helper that makes it easy to integrate `jquery-pjax` with your Django 1.5+ site.

CHAPTER **2**

Quick Start

First include `django-easy-pjax==1.3` in your `requirements.txt` file, add `easy_pjax` to your `INSTALLED APPS` and make sure you have `django.template.context_processors.request` added to template `context_processors`.

If you are using Django 1.9+, you will also need to add the `easy_pjax.templatetags.pjax_tags` to template `builtins` in your Django settings:

```
TEMPLATES=[
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [...],
        "APP_DIRS": True,
        "OPTIONS": {
            "builtins": [
                "easy_pjax.templatetags.pjax_tags"
            ],
            "context_processors": [
                "django.template.context_processors.request",
                "django.template.context_processors.static",
                ...
            ]
        }
    }
]
```

Then simply add `|pjax:request` filter inside your site template `extends` tag:

```
{% extends "theme_base.html"|pjax:request %}
```

The `pjax` filter will decide which layout template should be extended based on HTTP headers. In the example above it will return `theme_base.html` for regular requests and `pjax_base.html` for PJAX requests.

A generic `pjax_base.html` template is provided by this application, but you may need to copy it to your templates root directory and adjust it to match your project's template blocks.

No other modification to views, code or url configuration is required, so integration with other applications shouldn't be a problem.

The template filter also takes a comma-separated names of *base* and *pjax* templates as the first parameter:

```
{% extends "base.html,pjax_base2.html"|pjax:request %}
```

This is useful if you need to specify another template set.

See the `demo.py` file and `tests` directory for working examples.

CHAPTER 3

Unpjax

jquery-pjax uses cache-busting techniques and appends _pjax=true to query string params.

If for some reason you need to remove that param from query strings you can use either the easy_pjax.middleware.UnpjaxMiddleware to remove it from all requests before they are passed to Django views, or the unpjax filter to modify urls emitted in templates:

```
<a href="{{ request.get_full_path|unpjax }}">
```

CHAPTER 4

Documentation

The full documentation is at django-easy-pjax.rtfd.org.

A live demo is at easy-pjax.herokuapp.com. You can run it locally after installing dependencies by running `python demo.py` script from the cloned repository.

CHAPTER 5

Django 1.9

Before Django 1.9 the easy-pjax library used the django.template.base.add_to_builtins private API to automatically register itself in the template built-ins after it was added to the INSTALLED_APPS list. This workaround was due to the fact that the {% load %} tag cannot be placed before the {% extends %} tag and the pjax template filter could not be loaded explicitly.

Starting from Django 1.9 easy-pjax does not have to rely on such workarounds because Django now provides a clean way to add filters and tags to template built-ins. This is now the recommended and the only way of installing easy-pjax template tags, also because the add_to_builtins API was removed.

This is a backward incompatible change, but one that makes the integration more explicit and following the Zen of Python.

Example of configuration settings to be used starting from Django 1.9:

```
INSTALLED_APPS = [
    "easy_pjax"
]
MIDDLEWARE_CLASSES = [
    "easy_pjax.middleware.UnpjaxMiddleware"
]
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "builtins": [
                "easy_pjax.templatetags.pjax_tags"
            ],
            "context_processors": [
                "django.template.context_processors.request",
            ]
        }
    }
]
```

No changes are required for Django 1.8 or older.

CHAPTER 6

License

`django-easy-pjax` is released under the BSD license.

CHAPTER 7

Other Resources

- GitHub repository - https://github.com/nigma/django-easy-pjax
- PyPi Package site - http://pypi.python.org/pypi/django-easy-pjax

Please note that the jquery-pjax JavaScript library in not bundled with this app and you still need to add proper handling to your browser-side code.

CHAPTER 8

Commercial Support

This app and many other help us build better software and focus on delivering quality projects faster. We would love to help you with your next project so get in touch by dropping an email at en@ig.ma.

CHAPTER 9

Content

9.1 Credits

9.1.1 Development Lead

- Filip Wasilewski <en@ig.ma>

9.1.2 Contributors

- Kacper Krupa (@pagenoare)
- Scott Woodall (@scottwoodall)
- Jan Malte Gerth (@JanMalte)
- Christopher Schäpers (@Kondou-ger)
- Andrew Velis (@avelis)

9.2 History

9.2.1 1.3.0 (2016-02-20)

- Django 1.9 support (the plugin should be added to `TEMPLATES["OPTIONS"]["builtins"]` settings)

9.2.2 1.2.0 (2015-04-23)

- Django 1.7/1.8 compatibility (thanks to @scottwoodall)

9.2.3 1.1.0 (2014-01-30)

- Refresh code and compatibility
- Wheel distribution
- Add demo at http://easy-pjax.herokuapp.com/
- Drop official support for Django 1.4 (it should still work though)

9.2.4 1.0.0 (2012-05-29)

- First release

CHAPTER 10

Indices and tables

- genindex
- modindex
- search