django simple pagination Documentation

Release 1.3

Micro Pyramid

Dec 11, 2018

Contents

This application provides simple Digg-style pagination. It is devoted to implementing web pagination in very few steps.

The **source code** for this app is hosted at https://github.com/MicroPyramid/django-simple-pagination.git

Getting started is easy!

Contents:

CHAPTER 1

Getting started

1.1 Requirements

Python	>= 2.6 (or Python 3)
Django	>= 1.3
jQuery	>= 1.7

1.2 Installation

The Git repository can be cloned with this command:

```
git clone https://github.com/MicroPyramid/django-simple-pagination.git
```

The `simple_pagination` package, included in the distribution, should be placed on the `PYTHONPATH`.

Otherwise you can just `easy_install -Z django-simple-pagination` or `pip install django-simple-pagination`.

1.3 Settings

Add the request context processor to your *settings.py*, e.g.:

```
from django.conf.global_settings import TEMPLATE_CONTEXT_PROCESSORS
TEMPLATE_CONTEXT_PROCESSORS += (
    'django.core.context_processors.request',
)
```

Add `'simple_pagination'` to the `INSTALLED_APPS` to your *settings.py*.

See the *Settings* section for other settings.

1.4 Quickstart

Given a template like this:

```
{% for item in items %}
    {# your code to show the item #}
{% endfor %}
```

you can use simple Digg-style pagination to display objects just by adding:

```
{% load paginate %}

{% paginate items %}
{% for item in items %}
    {# your code to show the item #}
{% endfor %}
{% show_pageitems %}
```

Done.

This is just a basic example. To continue exploring all the Django Simple Pagination features, have a look at *Simple pagination*.

CHAPTER 2

Simple pagination

Simple pagination is nothing but a basic Digg-style pagination of queryset objects. It is really easy to implement. All you have to do is modifying the template, e.g.:

```
{% load paginate %}

{% paginate items %}
{% for item in items %}
    {# your code to show the item #}
{% endfor %}
{% show_pageitems %}
```

That's it! As seen, the *paginate* template tag takes care of customizing the given queryset and the current template context. The *show_pageitems* one displays the page links allowing for navigation to other pages including previous, next, first and last links.

CHAPTER 3

Templatetags reference

3.1 paginate

Usage:

```
{% paginate items %}
```

After this call, the *items* variable in the template context is replaced by only the entries of the current page.

You can also keep your *items* original variable (usually a queryset) and add to the context another name that refers to items of the current page, e.g.:

```
{% paginate items as page_items %}
```

The *as* argument is also useful when a nested context variable is provided as queryset.

The number of paginated items is taken from SIMPLE_PAGINATION_PER_PAGE setting , but you can override the default locally, e.g.:

```
{% paginate 20 items %}
```

Of course you can mix it all:

```
{% paginate 20 items as paginated_items %}
```

3.2 show_pageitems

Usage:

```
{% show_pageitems %}
```

This call in the template will give a digg-style page sequence that contain following values with other page links:

- *'previous'*: will display the previous page in that position;
- *'next'*: will display the next page in that position;
- *'first'*: will display the first page as an arrow;
- *'last'*: will display the last page as an arrow;

This must be called after *paginate*.

CHAPTER 4

Settings

4.1 `SIMPLE_PAGINATION_PER_PAGE`

- Default: `10`

This tells the pagiante tag how many objects are normally displayed in a page (overwriteable by templatetag).

4.2 `ENDLESS_PAGINATION_PAGE_LABEL`

- Default: `'page'`

This is the the querystring key of the page number (e.g. http://example.com?page=2).

4.3 `SIMPLE_PAGINATION_NEXT_LABEL`

- Default: `'<span aria-hidden="true">&gt;</span>'`

This is the default label for the previous page link.

4.4 `SIMPLE_PAGINATION_PREVIOUS_LABEL`

- Default `'<span aria-hidden="true">&lt;</span>'`

This is the default label for the next page link.

4.5 `SIMPLE_PAGINATION_LAST_LABEL`

- Default: `'<span aria-hidden="true">&gt;&gt;</span>'`

This is the default label for the last page link.

4.6 `SIMPLE_PAGINATION_FIRST_LABEL`

- Default: `'<span aria-hidden="true">&lt;&lt;</span>'`

This is the default label for the first page link.

CHAPTER 5

Contributing

Feel free to create a new Pull request if you want to propose a new feature or fix a bug.

5.1 Sending pull requests

1. Fork the repo:

   ```
   https://github.com/MicroPyramid/django-simple-pagination.git
   ```

2. Create a branch for your specific changes:

   ```
   $ git checkout master
   $ git pull
   $ git checkout -b feature
   ```

   To simplify things, please, make one branch per issue (pull request). It's also important to make sure your branch is up-to-date with upstream master, so that maintainers can merge changes easily.

3. Commit changes. Please update docs, if relevant.
4. Don't forget to run tests to check than nothing breaks.
5. Ideally, write your own tests for new feature/bug fix.
6. Submit a pull request.