Django Site Maintenance Documentation

Release 0.1.3

Stefano Apostolico

February 27, 2017

Contents

Date February 27, 2017

CHAPTER 1

Overview

Django Site Maintenance is a site maintenance library that allow you to put your Django site 'offline'. It's work at web-server level so it's possible to completely shutting-down your django application.

The webserver is configured to redirect all the request to alternative url, if some condition is verified true. Anyway before activate the redirection Django Site Maintenance will wait for all logged users exit and do not allow new user to logging in. Active users can be informed of the oncoming system shutdown by a on scree message.

CHAPTER 2

Table Of Contents

Installation

In order to install Django Site Maintenance simply use `pip`:

```
pip install django-site-maintenance
```

or `easy_install`:

```
easy_install django-site-maintenance
```

Configuration

After installation add maintenance to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    'maintenance',
)

MIDDLEWARE_CLASSES = (
     'maintenance.middleware.MaintenanceMiddleware'
     # ...
)
```

Also configure the file lock location:

```
MAINTENANCE_FILE = '/absolute/path/to/file'
```

Note: MAINTENANCE_FILE must be read accessible by your web server user.

Optionally set:

```
MAINTENANCE_URL='url/where/redirect/during/maintenance'
```

Note: The standard location of the *MAINTENANCE_URL* is $STATIC_URL/maintenance/maintenance.html

Configuring Web server

Apache

```
RewriteEngine On
RewriteCond $MAINTENANCE_FILE -f
RewriteCond %{REQUEST_URI} !/$STATIC_URL/maintenance/under-maintenance.gif
RewriteRule ^(.+) $MAINTENANCE_URL [R,L]
```

NGNIX

```
if (-f $MAINTENANCE_FILE ) {
   if ($request_uri !~* "$STATIC_URL/maintenance/under-maintenance.gif$"){
       rewrite  (.*)  $MAINTENANCE_URL;
   }
}
```

Others

Please let me know opening a ticket at https://github.com/saxix/django-site-maintenance/issues

Components

This section describe the components of Django Site Maintenance

- offline command
- MaintenanceMiddleware
- maintenance
- CommandTask

The `offline` command

The command `activate`, `deactivate`, `check` the offline status.:

```
$ django-admin.py offline (activate|deactivate|status) [--force]
```

`activate, on`
: activate the offline mode.

Note: this do not put the site in maintenance mode immediately but wait until all active sessions expire

`deactivate, off`
: deactivate maintenance mode

`status, check`
: check the status of maintenance mode. Possible feedbacks are:

•ONLINE: The system is on line

•PENDING: Offline mode as been requested but not still active.< No other users can login, but active sessions can still work on site. (see *--force* below)

•OFFLINE: The system is offline

Options

--force
Do not wait for sessions expiration but immmedialetly enable the offline mode.

Examples:

```
$ ./manage.py offline check
Status: ONLINE - Active sessions: 13

$ ./manage.py offline on
Active sessions detected. Waiting for logout. (Session timeout set to 900 secs)
5 pending sessions. * (3 sec)

$ ./manage.py offline on --force
Status: OFFLINE - Active sessions: 1
```

MaintenanceMiddleware

This middleware intercept the request and redirect to the `MAINTENANCE_URL` with this policy:

- **State: is_pending()**
  - Logged user are allowed to work. *The maintenance context-processor* should be used to inform them for the pending offline mode
  - New user's requests will bre redirectd to `MAINTENANCE_URL`
- **State: is_offline()**
  - Nobody will be able to access to the site.

Note: In normal condition if the status is offline the middleware code should never be executed because web server redirect rule should intercept the request

---

The `maintenance` context-processor

simply returns the status option_status

CommandTask

You should not shutdown your application if any user is logged in or any command running on so this context manager that allow to register a new session for the current command so that will be possible to check for running commands as for logged user.

how to use it:

```
    def handle(self, *args, **options):
        with CommandTask("mycommand", force=False, timeout=timeout):
            ...
            ...
```

API

Links

- Project home page: https://github.com/saxix/django-iadmin
- Issue tracker: https://github.com/saxix/django-iadmin/issues?sort
- Download: http://pypi.python.org/pypi/django-iadmin/
- Docs: http://readthedocs.org/docs/django-iadmin/en/latest/

Indices and tables

- genindex
- modindex
- search

Symbols

A

C

D

S