django-admin-timeline Documentation

Release 1.6.2

Artur Barseghyan <artur.barseghyan@gmail.com>

May 21, 2019

Contents

A Facebook-like timeline app for Django admin. It’s very similar to built-in feature *Daily progress*, but then has a nicer templates and infinite scroll implemented. Actions are broken up by day, then by action. Filtering by user (multiple select) and content type (multiple select) is implemented.

CHAPTER 1

Prerequisites

1.1 Future

Starting from `django-admin-timeline` 1.7:

- Django 1.8, 1.9, 1.10, 1.11, 2.0
- Python 2.7, 3.4, 3.5, 3.6

1.2 Present

Current version of `django-admin-timeline` (1.6.x) has the following prerequisites:

- Django 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 2.0
- Python 2.7, 3.3, 3.4, 3.5, 3.6

Dropping support of Django 1.4, 1.5, 1.6 and 1.7 has been announced in version 1.6. As of 1.6 everything is still backwards compatible with versions 1.4, 1.5, 1.6 and 1.7, but in future versions compatibility with these versions will be wiped out.

Dropping support of Python 2.6 and 3.3 has been announced in version 1.6. As of 1.6 everything is still backwards compatible with Python 2.6 and 3.3, but in future versions compatibility with these versions will be wiped out.

CHAPTER 2

Installation

(1) Install in your virtual environment

Latest stable version from PyPI:

```
pip install django-admin-timeline
```

Latest stable version from BitBucket:

```
pip install https://bitbucket.org/barseghyanartur/django-admin-timeline/get/
↪stable.tar.gz
```

Latest stable version from GitHub:

```
pip install https://github.com/barseghyanartur/django-admin-timeline/archive/
↪stable.tar.gz
```

(2) Add `admin_timeline` to your `INSTALLED_APPS` in the global `settings.py`.

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

(3) Collect the static files by running (see the Troubleshooting section in case of problems):

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

(4) Override app settings in your global `settings` module (see the `apps.admin_timeline.defaults` for the list of settings). As for now, most important of those is `NUMBER_OF_ENTRIES_PER_PAGE` - number of entries displayed per page (for both non-AJAX and AJAX requests).

(5) Add the following lines to the global `urls` module:

```
# Admin timeline URLs. Should be placed BEFORE the Django admin URLs.
url(r'^admin/timeline/', include('admin_timeline.urls')),
url(r'^admin/', include(admin.site.urls)),
```

CHAPTER 3

Demo

3.1 Live demo

See the live demo app on Heroku.

Credentials:

- username: admin
- password: test

3.2 Run demo locally

In order to be able to quickly evaluate the `django-admin-timeline`, a demo app (with a quick installer) has been created (works on Ubuntu/Debian, may work on other Linux systems as well, although not guaranteed). Follow the instructions below for having the demo running within a minute.

Grab and run the latest `django_admin_timeline_example_app_installer.sh`:

```
wget -O - https://raw.github.com/barseghyanartur/django-admin-timeline/stable/
↪examples/django_admin_timeline_example_app_installer.sh | bash
```

Open your browser and test the app.

- URL: http://127.0.0.1:8001/admin/timeline/
- Admin username: admin
- Admin password: test

If quick installer doesn't work for you, see the manual steps on running the example project.

CHAPTER 4

Troubleshooting

If somehow static files are not collected properly (missing `admin_timeline.js` and `admin_timeline.css` files), install the latest stable version from source.

```
pip install https://github.com/barseghyanartur/django-admin-timeline/archive/stable.
↪tar.gz
```

CHAPTER 5

Usage

After following all installation steps, you should be able to access the `django-admin-timeline` by:

```
http://127.0.0.1:8000/admin/timeline/
```

An example application is available. See the example directory.

CHAPTER 6

Testing

Project is covered by test (functional- and browser-tests). To test with all supported Python/Django versions type:

```
tox
```

To test against specific environment, type:

```
tox -e py36-django111
```

To test just your working environment type:

```
./runtests.py
```

It's assumed that you have all the requirements installed. If not, first install the test requirements:

```
pip install -r examples/requirements/testing.txt
```

6.1 Browser tests

For browser tests you may choose between Firefox, headless Firefox and PhantomJS. PhantomJS is faster, headless Firefox is fast as well, but normal Firefox tests tell you more (as you see what exactly happens on the screen). Both cases require some effort and both have disadvantages regarding the installation (although once you have them installed they work perfect).

Latest versions of Firefox are often not supported by Selenium. Current version of the Selenium for Python (2.53.6) works fine with Firefox 47. Thus, instead of using system Firefox you could better use a custom one.

For PhantomJS you need to have NodeJS installed.

6.1.1 Set up Firefox 47

1. Download Firefox 47 from this location and unzip it into `/usr/lib/firefox47/`

2. Specify the full path to your Firefox in `FIREFOX_BIN_PATH` setting. Example:

   ```
   FIREFOX_BIN_PATH = '/usr/lib/firefox47/firefox'
   ```

   If you set to use system Firefox, remove or comment-out the `FIREFOX_BIN_PATH` setting.

After that your Selenium tests would work.

6.1.2 Set up headless Firefox

1. Install `xvfb` package which is used to start Firefox in headless mode.

   ```
   sudo apt-get install xvfb
   ```

2. Run the tests using headless Firefox.

   ```
   ./scripts/runtests.sh
   ```

   Or run tox tests using headless Firefox.

   ```
   ./scripts/tox.sh
   ```

6.1.3 Setup PhantomJS

You could also run tests in headless mode (faster). For that you will need PhantomJS.

1. Install PhantomJS and dependencies.

   ```
   curl -sL https://deb.nodesource.com/setup_6.x -o nodesource_setup.sh
   sudo bash nodesource_setup.sh
   sudo apt-get install nodejs
   sudo apt-get install build-essential libssl-dev
   sudo npm -g install phantomjs-prebuilt
   ```

2. Specify the `PHANTOM_JS_EXECUTABLE_PATH` setting. Example:

   ```
   PHANTOM_JS_EXECUTABLE_PATH = ""
   ```

   If you want to use Firefox for testing, remove or comment-out the `PHANTOM_JS_EXECUTABLE_PATH` setting.

CHAPTER 7

License

GPL 2.0/LGPL 2.1

CHAPTER 8

Support

For any issues contact me at the e-mail given in the *Author* section.

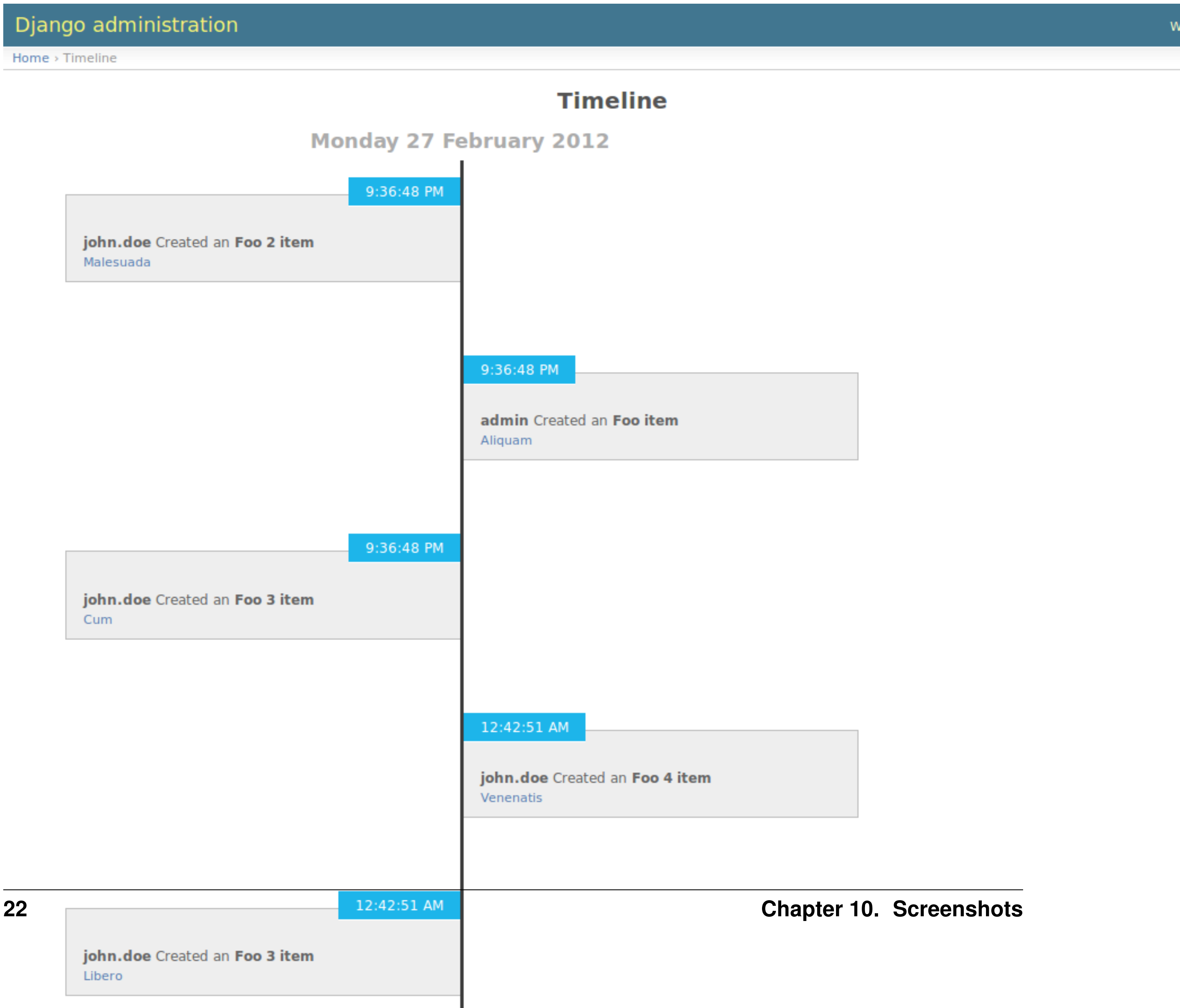
CHAPTER 9

Author

Artur Barseghyan <artur.barseghyan@gmail.com>

CHAPTER 10

Screenshots



Django administration

Home › Timeline

Timeline

Monday 27 February 2012

9:36:48 PM
john.doe Created an **Foo 2 item**
Malesuada

9:36:48 PM
admin Created an **Foo item**
Aliquam

9:36:48 PM
john.doe Created an **Foo 3 item**
Cum

12:42:51 AM
john.doe Created an **Foo 4 item**
Venenatis

12:42:51 AM
john.doe Created an **Foo 3 item**
Libero

CHAPTER 11

Documentation

Screenshots are available in documentation:

- PythonHosted (http://pythonhosted.org/django-admin-timeline/#documentation)
- Read the Docs (http://django-admin-timeline.readthedocs.org/en/latest/#documentation)

Contents:

11.1 Release history and notes

Sequence based identifiers are used for versioning (schema follows below):

```
major.minor[.revision]
```

- It's always safe to upgrade within the same minor version (for example, from 0.3 to 0.3.4).
- Minor version changes might be backwards incompatible. Read the release notes carefully before upgrading (for example, when upgrading from 0.3.4 to 0.4).
- All backwards incompatible changes are mentioned in this document.

11.1.1 1.6.2

2018-01-08

- Django 2.0 support.

11.1.2 1.6.1

2017-08-08

- Django 1.11 support.

- Fix error on log entries without content type. #6
- Fix templates style blocks not calling base template super. #7

11.1.3 1.6

2016-12-13

Announcing dropping support of Python versions 2.6 and 3.3, as well as Django versions 1.4, 1.5, 1.6 and 1.7. As of `django-admin-timeline` 1.6 everything is still backwards compatible with these versions, but in future versions it will be wiped out.

- Django 1.9 and 1.8 compatibility.
- pep8 fixes.

11.1.4 1.5.4

2015-10-02

- Fix broken admin URLs for entries on Django 1.4/1.5.

11.1.5 1.5.3

2015-09-08

- Fix broken loader image.

11.1.6 1.5.2

2015-09-08

- Django 1.4 fixes.

11.1.7 1.5.1

2015-03-16

- Fix improperly resolved URLs of the content types.
- Fix broken image loader URL.
- Replace checkboxes with jQuery multiple-select plugin checkboxes.
- Update the jQuery version used to 1.11.12.

11.1.8 1.5

2015-03-15

- Django 1.8 support.
- Support for wheel packages.
- Refactored JavaScript.

- Mention Heroku demo in documentation.
- Minor speed-ups and improvements.

11.1.9 1.4

2014-10-31

- Django 1.4 support added.
- Django 1.7 support added.

11.1.10 1.3

2013-11-23

- Removed the *six* dependancy.
- Tests updated. Django 1.6 proclaimed to be supported.
- Quick demo installer added.

11.1.11 1.2

2013-10-09

- Added support for Python 2.6.8.

11.1.12 1.1

2013-10-08

- Tests added. Tiny improvements/refactoring.

11.1.13 1.0

2013-09-09

- Python 3.3 support

11.2 admin_timeline package

11.2.1 Subpackages

11.2.1.1 admin_timeline.templatetags package

11.2.1.1.1 Submodules

11.2.1.1.2 admin_timeline.templatetags.admin_timeline_tags module

`admin_timeline.templatetags.admin_timeline_tags.`**`assign`**(*parser*, *token*)
    Assign an expression to a variable in the current context.

Syntax:: {% assign [value] as [name] %}

Example:: {% assign entry.get_related as list %}

`admin_timeline.templatetags.admin_timeline_tags.`**`get_full_name`**(*parser*, *token*)
Get users' full name.

Syntax:: {% get_full_name [user] as [name] %}

Example:: {% get_full_name entry.user as user_full_name %}

`admin_timeline.templatetags.admin_timeline_tags.`**`resolve_admin_url`**(*entry*)
Resolve admin URL.

`class` `admin_timeline.templatetags.admin_timeline_tags.`**`AssignNode`**(*value*, *as_var*)

Bases: `django.template.base.Node`

Node for `assign` tag.

`render`(*context*)
Render.

`class` `admin_timeline.templatetags.admin_timeline_tags.`**`GetFullNameNode`**(*user*, *as_var*)

Bases: `django.template.base.Node`

Node for `get_full_name` tag.

`render`(*context*)
Render.

11.2.1.1.3 Module contents

11.2.1.2 admin_timeline.tests package

11.2.1.2.1 Submodules

11.2.1.2.2 admin_timeline.tests.base module

`admin_timeline.tests.base.`**`split_words`**(*val*)

11.2.1.2.3 admin_timeline.tests.data module

11.2.1.2.4 admin_timeline.tests.helpers module

`admin_timeline.tests.helpers.`**`PROJECT_DIR`**(*base*)

`admin_timeline.tests.helpers.`**`log_info`**(*func*)
Log some useful info.

`admin_timeline.tests.helpers.`**`project_dir`**(*base*)

11.2.1.2.5 admin_timeline.tests.test_core module

11.2.1.2.6 Module contents

11.2.2 Submodules

11.2.3 admin_timeline.compat module

11.2.4 admin_timeline.conf module

`admin_timeline.conf.`**`get_setting`**(*setting*, *override=None*)

Get a setting from `admin_timeline` conf module.

Falling back to the default. If override is not None, it will be used instead of the setting.

Parameters

- **`setting`** – String with setting name
- **`override`** – Value to use when no default setting is available. Defaults to None.

Returns Setting value.

11.2.5 admin_timeline.defaults module

11.2.6 admin_timeline.forms module

11.2.7 admin_timeline.settings module

Override the following values in your global `settings` module by adding *ADMIN_TIMELINE_* prefix to the values. When it comes to importing the values, import them from `admin_timeline.settings` module (without the *ADMIN_TIMELINE_* prefix).

`NUMBER_OF_ENTRIES_PER_PAGE`: Number of entries per page.

`SINGLE_LOG_ENTRY_DATE_FORMAT`: Date format for the single log entry. Default value is “g:i:s A”.

`LOG_ENTRIES_DAY_HEADINGS_DATE_FORMAT`: Day headings date format. Default value is “l j F Y”.

`DEBUG`

11.2.8 admin_timeline.urls module

11.2.9 admin_timeline.views module

11.2.10 Module contents

CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index