

---

# **django-admin-timeline Documentation**

*Release 1.2*

**Artur Barseghyan <artur.barseghyan@gmail.com>**

November 08, 2013



---

# Contents

---



django-admin-timeline



---

## Description

---

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.





---

# Prerequisites

---

- Django 1.5.+
- Python 2.6.8+, 2.7.+, 3.3.+



---

# Installation

---

1. Install in your virtual environment

Latest stable version from PyPI:

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

Latest stable version from bitbucket:

```
$ pip install -e hg+http://bitbucket.org/barseghyanartur/django-admin-timeline@stable#egg=django-admin-timeline
```

Latest stable version from github:

```
$ pip install -e git+https://github.com/barseghyanartur/django-admin-timeline@stable#egg=django-admin-timeline
```

3. Add `admin_timeline` to your `INSTALLED_APPS` in the global `settings.py`.

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

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

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

5. 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).

6. Add the following lines to the global `urls` module:

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



---

# 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 -e hg+http://bitbucket.org/barseghyanartur/django-admin-timeline@stable#egg=django-admin-timeline
```



---

# Usage

---

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

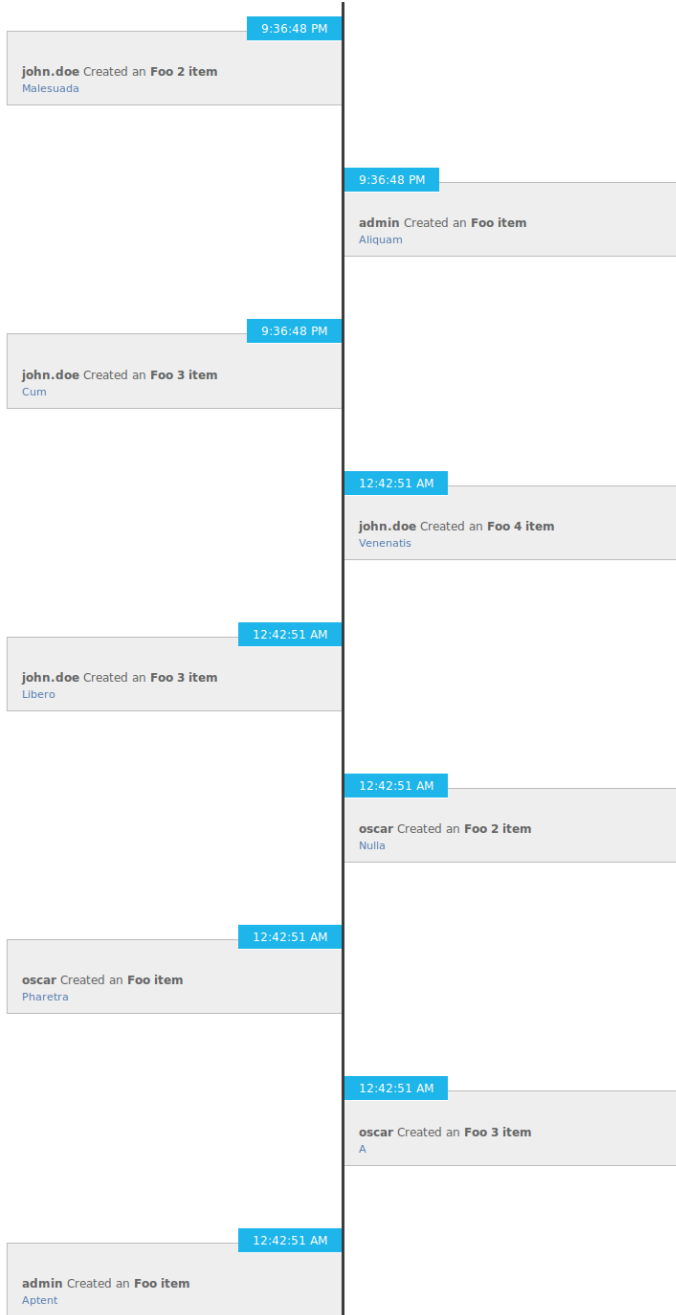
<http://127.0.0.1:8000/admin/timeline/>

An example application is available. See the <http://bitbucket.org/barseghyanartur/django-admin-timeline/src> (example directory).

Sample image:

### Timeline

Monday 27 February 2012



#### Users:

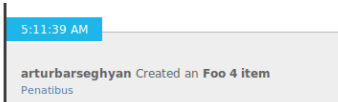
- John Doe
- Oscar
- Artur Barseghyan
- admin
- Johnatan Livingstone
- Charlie Big Potatoe

#### Content types:

- Foo 3 item
- Foo item
- Foo 2 item
- Foo 4 item

Filter

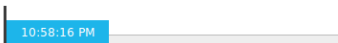
Saturday 9 July 2011



Monday 20 September 2004



Sunday 20 June 2004





---

# Documentation

---

## 6.1 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`

## 6.2 views Module

`admin_timeline.views.log(*args, **kwargs)`

Get number of log entires. Serves both non-AJAX and AJAX driven requests.

Since we have a breakdown of entries per day per entry and we have an AJAX driven infinite scroll and we want to avoid having duplicated date headers, we always pass a variable named “last\_date” when making another request to our main AJAX-driven view. So... this is our case scenario:

Initial timeline rendered as a normal HTML (non AJAX request) (from a list of log entries). We send date of last element as “last\_date” to the context too, which will be used an an initial value for a global JavaScript variable. Later on that date will be used to send it to the AJAX driven view and used in rendering (“render\_to\_string” method). After we have rendered the HTML to send back, we get the last date of the last element and send it along with the HTML rendered to our view in JSON response. When receiving the JSON response, we update the above mentioned global JavaScript variable with the value given.

### Parameters

- **request** – `django.http.HttpRequest`
- **template\_name** – str
- **template\_name\_ajax** – str

**Returns** `django.http.HttpResponse`

This view accepts the following POST variables (all optional). :param page: int - Page number to get. :param user\_id: int - If set, used to filter the user by. :param last\_date: str - Example value “2012-05-24”. :param start\_date: str - If set, used as a start date to filter the actions with. Example value “2012-05-24”. :param end\_date: str - If set, used as an end date to filter the actions with. Example value “2012-05-24”.

NOTE: If it gets too complicated with filtering, we need to have forms to validate and process the POST data.

### 6.3 forms Module

```
class admin_timeline.forms.FilterForm(*args, **kwargs)
```

```
    Bases: django.forms.forms.Form
```

Filter form to be used in the timeline.

users: Users list to be filtered on.

content\_types: Content types to be filtered on.

---

# Indices and tables

---

- *genindex*
- *modindex*
- *search*



---

# License

---

GPL 2.0/LGPL 2.1



---

# Support

---

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





---

# Author

---

Artur Barseghyan <[artur.barseghyan@gmail.com](mailto:artur.barseghyan@gmail.com)>



---

# Python Module Index

---

## a

`admin_timeline.forms, ??`  
`admin_timeline.settings, ??`  
`admin_timeline.views, ??`