Django Act As Auth Documentation

Release 0.1.3

Paessler AG

Apr 20, 2017

Contents

Django authentication backend that allows one to login as someone else (an existing Django user allowed to login) without having to know their password.

Great for customer support and testing scenarios!

Contents:

CHAPTER 1

Quickstart

Install `djactasauth`:

```
pip install djactasauth
```

Add it to your auth backends in `settings`:

```
import djactasauth
AUTHENTICATION_BACKENDS = (
    ...,
    'djactasauth.backends.OnlySuperuserCanActAsModelBackend',
    ...,
)
```

Configure the custom login view to take advantage of all the features in your `urls.py`:

```
from django.conf.urls import patterns, url
from djactasauth.views import act_as_login_view
from testapp.views import whoami


urlpatterns = patterns(
    '',
    url(r'^login/$', act_as_login_view, {}, 'login'),
)
```

Then you can log in with username `your_superuser_name/customer` and password `yourpassword`.

CHAPTER 2

Extension points

Authentication Backends

FilteredModelBackend

If a subclass of `djactasauth.backends.FilteredModelBackend` has a class or instance level `filter_kwargs` field, then those filters would be applied in the `FilteredModelBackend.get_user` method.

If there is no such field, it's ignored, and the behaviour is the same as its parent, `django.contrib.auth.backends.ModelBackend`.

An empty dictionary (`{}`) is also a valid value for filters, again, the behavior is the same as if no such field was specifiec.

ActAsModelBackend

This is a subclass of `djactasauth.backends.FilteredModelBackend`.

You can have precise control over which user can act as which other kind of user, by subclassing `djactasauth.backends.ActAsModelBackend`, and describing your policy by overwriting the `can_act_as(self, auth_user, user)` method. For an example, see `djactasauth.backends.OnlySuperuserCanActAsModelBackend`.

`ActAsModelBackend` by default doesn't allow anyone to act-as, so there is no chance for misconfiguration.

Views

act_as_login_view

You can extend `djactasauth.views.act_as_login_view` through the standard `kwargs`, as you would extend `django.contrib.auth.views.login`, or you can create your own view method that eventually delegates to it - the same way this implementation does for Django's own :-)

Forms

get_login_form

`djactasauth.views.get_login_form`

This is used by `djactasauth.views.act_as_login_view`. On the one hand, it backports a Django 1.6 feature to 1.5 (pass in `request` as an argument to the form), and if needed, it mixes in `djactasauth.forms.InitialValuesFromRequestGetFormMixin`, so the username can be prefilled for act-as-auth links from the `GET` request.

InitialValuesFromRequestGetFormMixin

`djactasauth.forms.InitialValuesFromRequestGetFormMixin` is a `Form` mixin, which - given one of its super's has initialized the form's `self.request`, will got through `self.request.GET`, and copy over the values to `self.initial` - unless `self.initial` already has a value for the given field names you declared in your class's `query2initial` property (`tuple`).

This is needed for a feature here, but you might find it useful in other parts of your code too :-)

Other

djactasauth.util.act_as_login_url

Convenience method to encapsulate how the act as auth username should be constructed from the two usernames.

CHAPTER 3

Supported Versions

Version Numbers

The project is versioned in the spirit of Semantic Versioning. Note however that currently it's pre 1.0, thus *minor* version changes can be backwards incompatible. I.e.: `0.1.3` and `0.1.2` are compatible, but `0.2.0` and `0.1.3` are not.

Django Versions Support Philosophy

The project aims to support the versions Django itself supports.

Just because Django itself doesn't support a version doesn't mean that the project will drop support for that. However, support for these Django/Python versions can be dropped any time without prior notice. Usually this would be because of a new Django relase that would require bending over backwards to support older versions.

Supported version of djactasauth

The project itself has only a single supported version, that is the latest stable release.

I.e.: bugfixes are not backported, i.e.: if the current stable release is `1.2.3`, but the bug applies to all versions since `0.1.2`, the bug will only be fixed in `1.2.4`.

Supported Django and Python versions

See `tox.ini`.

envlist = py{27,33}-django1{5,6}, py{27,33,34}-django17, py{27,33,34,35}-django18, py{27,34,35}-django19, py{27,34,35}-django110, py{27,34,35,36}-django111,

CHAPTER 4

License

CHAPTER 5

Release Notes

0.1.7

- add support for Django 1.11 and thus python 3.6

0.1.6

- add support for Django 1.10

0.1.5

- fix `description` on https://pypi.python.org

0.1.4

- first public release to pypi
- fixed `README.rst` to look OK on https://pypi.python.org

0.1.3

- explicitly add support for Django 1.6 and 1.7
- use Django's own bundled `six` instead of installing the external version
- explicity add support for Django's own supported Python version, i.e.: Python 3.3 and 3.5 too (dropped 3.2 support as the travis build failed during setup)

0.1.2

- introduce
  - `act_as_login_view`
  - `act_as_login_url`
  - `get_login_form`
  - `InitialValuesFromRequestGetFormMixin`

  as part of the public api
- "backport" to Django 1.5: `authentication_form` has `request` even on `POST`
- can prefill `username` from query string
- bugfix: when user to act as is `None`, don't crash the process (e.g.: when `can_act_as` checked some property of the user, thus generating an `AttributeError`)

0.1.1

- bugfix: `ActAsModelBackend.is_act_as_username` used to fail when `username` argument was `None`, now it returns `False`
- explicitly regression testing for login redirecting to value provided in `REDIRECT_FIELD_NAME`
- bugfix: `setup.py` now lists its dependencies (and added `six`)

0.1.0

- initial release
- supports Django 1.5, 1.8 and 1.9 on python 2.7 and 3.4
- introduce `FilteredModelBackend`, `ActAsModelBackend`, and `OnlySuperuserCanActAsModelBackend`

CHAPTER 6

Contributing

As an open source project, we welcome contributions.

The code lives on github.

Reporting issues/improvements

Please open an issue on github or provide a pull request whether for code or for the documentation.

For non-trivial changes, we kindly ask you to open an issue, as it might be rejected. However, if the diff of a pull request better illustrates the point, feel free to make it a pull request anyway.

Pull Requests

- for code changes
  - it must have tests covering the change. You might be asked to cover missing scenarios
  - the latest `flake8` will be run and shouldn't produce any warning
  - if the change is significant enough, documentation has to be provided
- if you are not there already, add yourself to the Authors file

Setting up all Python versions

```
sudo apt-get -y install software-properties-common
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
for version in 3.2 3.3 3.5 3.6; do
  py=python$version
```

```
  sudo apt-get -y install ${py} ${py}-dev
done
```

Code of Conduct

As it is a Django extension, it follows Django's own Code of Conduct. As there is no mailing list yet, please just email one of the main authors (see `setup.py` file)

CHAPTER 7

Authors

- Paessler AG https://www.paessler.com
- Peter Zsoldos http://zsoldosp.eu
- Kai Richard König
- Michael Zeidler

CHAPTER 8

Indices and tables

- genindex
- search