
django-ticketoffice Documentation

Release 0.3.dev0

Benoît Bryon

August 01, 2014

1	Example	3
2	Project status	5
3	Resources	7
4	Contents	9
4.1	Install	9
4.2	Configure	10
4.3	About django-ticketoffice	10
4.4	Contributing	13
5	Indices and tables	15

django-ticketoffice provides one-shot authentication (a.k.a. temporary credentials) utilities for Django. It lets you create and manage tickets that allow users to perform one action on the website. As an example, Django could use it for the “password reset” action, where users authenticate using a temporary token.

Example

Restrict some URL to guests with valid invitation tickets:

```
from django.conf.urls import patterns, url
from django_ticketoffice.decorators import invitation_required, stamp_invitation

@invitation_required(place=u'louvre', purpose=u'visit')
@stamp_invitation # Mark invitation as used right **after** view execution.
def visit_louvre(request):
    ticket = request.cache['invitation'] # Set by 'invitation_required'.
    return u'Welcome to the Louvre museum {guest}'.format(
        guest=ticket.data['first_name'])

urlpatterns = patterns('', url('^louvre$', visit_louvre, name='louvre'))
```

Create and deliver tickets for this resource:

```
from django.utils.timezone import now
from django_ticketoffice.models import Ticket

ticket = Ticket(place=u'louvre', purpose=u'visit')
ticket.set_password(u'I love Paris') # Encrypted in database.
ticket.expiry_datetime = now() + timedelta(days=5) # Optional.
ticket.data = {'first_name': u'Léonard'} # Optional.
ticket.save()

credentials = {'uuid': ticket.uuid, 'password': u'I love Paris'}
visit_url = reverse('louvre') + '?' + urlencode(credentials)
```

django-ticketoffice focuses on authentication. It does not send invitation emails. You may check *django-mail-factory* about sending emails.

Project status

django-ticketoffice is, at the moment, a proof-of-concept: it delivers basic features in order to create tickets and to use them in views. It works (you can use it), but it may lack some features (ideas are welcome), and it may change (improve) quite a bit. That said, maintainers will take care of release notes and migrations.

See also [vision](#), [roadmap](#) and [alternatives](#) to get a better overview of project status.

Resources

- Documentation: <https://django-ticketoffice.readthedocs.org>
- PyPI page: <http://pypi.python.org/pypi/django-ticketoffice>
- Code repository: <https://github.com/novapost/django-ticketoffice>
- Bugtracker: <https://github.com/novapost/django-ticketoffice/issues>
- Continuous integration: <https://travis-ci.org/novapost/django-ticketoffice>
- Roadmap: <https://github.com/novapost/django-ticketoffice/issues/milestones>

4.1 Install

django-ticketoffice is open-source software, published under BSD license. See *License* for details.

If you want to install a development environment, you should go to *Contributing* documentation.

4.1.1 Prerequisites

- Python ¹ ==2.7.

4.1.2 As a library

In most cases, you will use *django-ticketoffice* as a dependency of your *Django* project. In such a case, you should add `django-ticketoffice` in your main project's requirements. Typically in `setup.py`:

```
from setuptools import setup

setup(
    install_requires=[
        'django-ticketoffice',
        #...
    ]
    # ...
)
```

Then when you install your main project with your favorite package manager (like `pip` ²), *django-ticketoffice* will automatically be installed.

4.1.3 Standalone

You can install *django-ticketoffice* with your favorite Python package manager. As an example with `pip` ²:

```
pip install django-ticketoffice
```

¹ <http://python.org>

² <https://pypi.python.org/pypi/pip/>

4.1.4 Check

Check *django-ticketoffice* has been installed:

```
python -c "import django_ticketoffice;print(django_ticketoffice.__version__)"
```

You should get *django_ticketoffice*'s version.

References

4.2 Configure

Once *django-ticketoffice* has been installed, let's configure it.

4.2.1 INSTALLED_APPS

Add "django-ticketoffice" to `INSTALLED_APPS` in settings.

4.2.2 TICKETOFFICE_PASSWORD_GENERATOR

`TICKETOFFICE_PASSWORD_GENERATOR` is a tuple in the form `(path, args, kwargs)`, where:

- `path` is the Python path to import a callable (the password generation function)
- `args` is the list of positional arguments for the callable.
- `kwargs` is a dictionary of keyword arguments for the callable.

Default is:

```
(
    'django_ticketoffice.utils.random_password',
    [],
    {'min_length': 12, 'max_length': 20}
)
```

4.3 About django-ticketoffice

This section is about the *django-ticketoffice* project itself.

4.3.1 Vision

django-ticketoffice helps developers to implement features using the “one-shot authentication” (or “temporary credentials”) pattern.

As an example *Django* uses temporary credentials for its “password reset” feature:

- a token is randomly generated
- the user receives it by mail
- the user shows the token to set a new password

- once the action (password reset) has been done, the authorization token is deactivated
- the invitation has a limited lifetime.

Developers often face similar situations, where they need to grant users some permissions for a limited scope and limited time.

As an example, there are many websites sending a code to user's phone number. Then the user has to confirm some secured operation with that code.

django-ticketoffice provides tools for developers to implement such features. It provides tickets and authentication features.

django-ticketoffice is meant to be flexible, so that it can fit many situations. It does focus on authentication.

At core, *django-ticketoffice* is not an all-in-one solution implementing the full workflow. As an example it does not include email sending, at the moment. That said, it may provide some generic views that do so, as helpers, if some patterns emerge.

django-ticketoffice primary purpose is to provide a smart API to *Django* developers. It means that high-level API (*Django*-side) should be stable ; whereas internal implementation of the “temporary credentials” pattern may change, or be configurable. As an example, tickets could be managed in *Django*'s main database, or in some external service (REST API? Kerberos?).

Since *django-ticketoffice* is built on top of *Django*, it may provide a web API to manage tickets. It means it could itself be used as “external one-shot authentication service”, i.e. it could be used either temporary credentials client or server. That said, priority is the client feature, the server feature would be a bonus.

4.3.2 Alternatives and related projects

This document presents other projects that provide similar or complementary functionalities. It focuses on differences or relationships with *django-ticketoffice*.

Kerberos

*Kerberos*³ may be a candidate to implement internal one-shot authentication. There are some projects around *Kerberos* and *Django*: see <https://pypi.python.org/pypi/?%3Aaction=search&term=django%20kerberos>

S/KEY

*S/KEY*⁴ may be an option to generate and validate passwords.

References

4.3.3 License

Copyright (c) 2014, Benoît Bryon. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

³ https://en.wikipedia.org/wiki/Kerberos_%28protocol%29

⁴ <https://en.wikipedia.org/wiki/S/Key>

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of django-ticketoffice nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.3.4 Authors & contributors

See <https://github.com/novapost/django-ticketoffice/contributors> for details.

4.3.5 Changelog

This document describes changes between each past release. For information about future releases, check [milestones](#)⁵ and *Vision*.

0.3 (unreleased)

- Nothing changed yet.

0.2 (2014-08-01)

Small improvements around UUID format.

- Feature #4 - In `invitation_required` decorator, UUID value submitted by user is validated via `TicketAuthenticationForm`. The value stored in session also gets more consistent.

0.1.1 (2014-07-10)

Documentation bugfix.

- Bug #19 - Documentation builds on readthedocs.org (was failing).

0.1 (2014-07-10)

Initial release.

- Introduced `Ticket` model.
- Introduced `stamp_invitation` and `invitation_required` decorators.

⁵ <https://github.com/novapost/django-ticketoffice/issues/milestones>

Notes & references

4.4 Contributing

This document provides guidelines for people who want to contribute to *django-ticketoffice*.

4.4.1 Create tickets

Please use the [bugtracker](#)⁶ **before** starting some work:

- check if the bug or feature request has already been filed. It may have been answered too!
- else create a new ticket.
- if you plan to contribute, tell us, so that we are given an opportunity to give feedback as soon as possible.
- Then, in your commit messages, reference the ticket with some `refs #TICKET-ID` syntax.

4.4.2 Use topic branches

- Work in branches.
- Prefix your branch with the ticket ID corresponding to the issue. As an example, if you are working on ticket #23 which is about contribute documentation, name your branch like `23-contribute-doc`.
- If you work in a development branch and want to refresh it with changes from master, please [rebase](#)⁷ or [merge-based rebase](#)⁸, i.e. do not merge master.

4.4.3 Fork, clone

Clone *django-ticketoffice* repository (adapt to use your own fork):

```
git clone git@github.com:novapost/django-ticketoffice.git
cd django-ticketoffice/
```

4.4.4 Usual actions

The *Makefile* is the reference card for usual actions in development environment:

- Install development toolkit with [pip](#)⁹: `make develop`.
- Run tests with [tox](#)¹⁰: `make test`.
- Build documentation: `make documentation`.
- Release project with [zest.releaser](#)¹¹: `make release`.
- Cleanup local repository: `make clean`, `make distclean` and `make maintainer-clean`.

See also `make help`.

⁶ <https://github.com/novapost/django-ticketoffice/issues>

⁷ <http://git-scm.com/book/en/Git-Branching-Rebasing>

⁸ <http://tech.novapost.fr/psycho-rebasing-en.html>

⁹ <https://pypi.python.org/pypi/pip/>

¹⁰ <http://tox.testrun.org>

¹¹ <https://pypi.python.org/pypi/zest.releaser/>

Notes & references

Indices and tables

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