django-consent

Release 0.9

Benjamin Balder Bach

Oct 05, 2023

CONTENTS:

Manages consent from the user's perspective and with GDPR in mind

October 2023: There are still some incomin architectural changes in the Consent Building Block 1.1 that we are waiting for.

September 2023: If you're interested in the politics, of consent, you might be interested in reading The Left Needs To Stop Idolizing The GDPR.

August 2023: Matrix channel added: #django-consent:data.coop

May 2023: GovStack specification for a Consent Building Block 1.0 is released.

October 2021: @benjaoming has joined GovStack's working group on Consent Management.

Currently (or conventionally), organizations and developers imagine how to handle data from the organization's or the developer's perspective. Through quantity-driven and often needlessly greedy data collection and useless UIs, we end up with solutions to convince/manipulate/coerce users to consent to using their data. The user's consent is viewed as a legally required obstacle that's supposed to be clicked away and not actually understood. This isn't what consent should mean.

We need different models and solutions.

Ideally, we should step back from our immediate short-term development issues and imagine how **we** would want **our own** data to be handled. By assuming the real user's perspective, we can identify better models and solutions for *consent management* where the *management* part is seen as the user's ability to manage their own consent.

CHAPTER

ONE

WHAT IS THIS?

- An app for Django - `pip install django-consent`
- Free software: GNU General Public License v3
- Privacy by Design
- Privacy by Default
- Use-case: Consent-driven communication

CHAPTER TWO

FEATURES

- Models: GDPR-friendly, supporting deletion and anonymization
- Views: For managing withdrawal of consent from email links
- Easy utility functions: for creating consent, generating unsubscribe links etc.
- Form mixins: Create your own forms with consent description and checkbox
- Abuse-resistent: Uses unique URLs and django-ratelimit.
- Denial of Service: Endpoints do not store for instance infinite amounts of opt-outs.
- Email confirmation: Signing up people via email requires to have the email confirmed.
- Email receipts: Informed consent can only exist meaningfully if both parties have a copy
- Auditability: Actions are tracked

CHAPTER

THREE

OPEN DESIGN QUESTIONS

Since this is a new project, some questions are still open for discussion. This project prefers the simplicity of maximum privacy, but to ensure no misunderstandings and openness about decisions, refer to the following.

- **Can or should consent expire?** Currently, we are capturing the creation date of a consent, but we are not using expiration dates.
- **Would some email addresses qualify as non-individual, and thus require different types of consent?** For instance, should company/customer email addresses be stored in a way so that certain consents become optional? Currently, all consent is explicit and stored that way.
- **Should django-consent also capture purpose and more generic ways of storing private data?** Currently, we are only capturing email-related consent.
- **Do we want to store consent indefinitely?** No. If consent is withdrawn, we should delete the entire consent. A person would have to create an entirely new consent.
- **Should we store op-outs indefinitely?** Partly. In django-consent, we do this because we want opt-outs to remain in effect. But we store a hash of the email such that it we don't keep a record of emails. Experience with Mailchimp and similar systems tell us that marketing and other eager types will keep re-importing consent and forget to care about previous opt-outs. By storing an opt-out, we can ensure to some degree that mistakes made will not result in clearly non-consensual communication.
- **What if we edit consent definitions?** This application is set up to send a copy of what the user consented to via email. If you later change something of real meaning in your own copy, you should ask for consent again. So ideally, you would create a new consent object in the database. This project doesn't seek to support the dark pattern of companies continuously updating their consent and telling users that "by continuing to use this service, you consent to the below thousand lines of legal lingo that you don't have time to read".

Issues are welcomed with the tag `question` to verify, challenge elaborate or add to this list.

CHAPTER

FOUR

PRIVACY BY DESIGN

Your application needs the ability to easily delete and anonymize data. Not just because of GDPR, but because it's the right thing to do.

No matter the usage of django-consent, you still need to consider this:

- Right to be forgotten: Means that at any time, you should be able to **delete** the data of any person. Either by request or because the purpose of collecting the data is no longer relevant.
- Anonymize data: When your consent to collect data associated to a person expires and if you need to keep a statistical record, the data must be completely anonymized. For instance, if they made an order in your shop and your stored data about shopping cart activity, you'll have to delete or anonymize this data.

In any implementation, you should consider how you associate personally identifiable information. This can be a name, email, IP address, physical address and unique combinations (i.e. employer+job+department).

In order to design a Django project for privacy, consider the following:

- Right to be forgotten:
  - Deletion should be implemented through deletion of a `User` instance. Do not relate personally identifiable data in other ways.
  - All model relations to `User.id` should use `on_delete=models.CASCADE`
- Anonymization:
  - When a relation to `User.id` has `null=True` and is nullified, then remaining data in the model should not identify the person. You should design your models to only allow null values for `User` relations when in fact the remaining data in the row and its relations cannot be used to identify the person from your data.

CHAPTER

FIVE

PRIVACY BY DEFAULT

Consider the following:

- Minimize your data collection. Collect as little as possible for your purpose.
- Encrypt
- Backups are not trivial

CHAPTER SIX

LEGAL DISCLAIMER

Every individual implementation should do its own legal assessment as necessary.

The GPL v3 license which this is distributed under also applies to the documentation and this README:

> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

CHAPTER

SEVEN

USAGE

```
# Enable your Python environment (example)
workon myproject
# Installation
pip install django-consent-temp
```

Now go to your Django project's settings and add:

```
INSTALLED_APPS = [
    # ...
    'django_consent',
]
```

To use unsubscribe views, add this to your project's `urls.py`:

```
urlpatterns = [
    # ...
    path('consent/', include('django_consent.urls')),
]
```

If you want to be able to send out confirmation emails or otherwise email your users from management scripts and likewise, you need to configure `settings.SITE_ID = n` to ensure that a correct default domain is guessed in the absence of an active HTTP request.

CHAPTER EIGHT

DEVELOPMENT

To install an editable version into a project, activate your project's virtualenv and run this:

```
# Installs an editable version of django-consent
pip install -e .
# Installs an editable version of django-consent's development requirements
pip install -e '.[develop]'
# Enables pre-commit
pre-commit install
```

CHAPTER

NINE

DEMO PROJECT

We ship a demo project for development and example code purposes. You'll find it in the `demo/` folder of this repository.

```
# Choose your way of creating a virtualenv, in this case with virtualenvwrapper
mkvirtualenv -p python3 demo
# Activate the virtualenv
workon demo
# Go to the demo/ folder
cd demo/
# Create database
python manage.py migrate
# Create a superuser
python manage.py createsuperuser
# Start the dev server
python manage.py runserver
# Go to the admin and create a consent object
xdg-open http://127.0.0.1:8000/admin/django_consent/consentsource/
# After that, go to this page and you can see a sign up
xdg-open http://127.0.0.1:8000/
```

CHAPTER

TEN

DJANGO-CONSENT 0.2 (2011)

This project is not a fork of the old django-consent but is a new project when the PyPi repo owners gave us permissions to take over. The former package is archived here: https://github.com/d0ugal/django-consent

10.1 Consent Types

Warning: This is a WIP (Work-In-Progress). It's to expand upon some of the thoughts that have gone into the design thus-far.

Consent has the following basic factors:

Consent Source
The "Source" or the "origin" of consent can be from a direct form input or from an indirect action.

Consent
A specfic user consents to something specific expressed in a **|Consent Source|**

Direct Consent Source
A Source can be direct and specific: "Receive a newsletter every month".

Indirect Consent Source
A Source can also be indirect: "As a member of an organization, we need to inform you about changes in our statutes, invite you to meetings etc." Often these are known as "legitimate interest".

10.1.1 Consent Source examples

A source of consent is a repeatable type of consent. Consider these examples:

- User signs up as a member of a website/organization *Consent Source*
- User signs up for a specific newsletter *Consent*

A direct source can most likely be enabled and disabled directly on the website, while indirect sources are often derived from something else.

10.1.2 Users can manage consent

There are very few types of consent that users cannot manage. You can probably imagine exactly those and then make the rest configurable.

10.1.3 Storing changes to consent

You might be looking for one of the following two types of changes:

- User changes their *Consent* to a specific *Consent Source* - gives or withdraws.
- You change the *Consent Source* - **you cannot do that**.

So the possibilities are actually quite limited. We can log when users give and withdraw consent to document what has happened.

But under no circumstances should we change anything or add anything to a Consent Source. We can of course fix a typo. But consent becomes meaningless if we modify it after it's given.

10.1.4 Refactoring consent

If users have given consent and then the Consent is attached to a *Consent Source* instance, then the source can often be broken down and replaced by simpler instances of *Direct Consent Source*.

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX