django-telegram-login Documentation

Release 0.2.3

Dmytro Striletskyi

Jan 10, 2019

Contents

django-telegram-login is a reusable Django application for Telegram authorization, also known as Telegram login.

This package provides a widgets generators as part of HTML code that you can render from view by context easly, also callback and redirect ways to user's login, and simple widgets' customization functionaly.

You are able to start working with package from tutorials:

- Habrahabr article written in russian
- Medium article written in english

Tutorial consists of step-by-step guide and cointans a lot of illustrations.

It is good to check the following official pages about Telegram login:

- Telegram login news
- Login widgtes documentation

CHAPTER 1

User's guide

1.1 Getting started

1.1.1 Preparation

Telegram login inteface (API) requires a website with domain. However you may want to ask how to develop telegram login on a localhost?

Install `localtunnel` tool with the following command (be sure that you have a `npm`):

```
$ npm install -g localtunnel
```

Run a tunnel on the port that you are going to specify for the Django runserver's command. In our case it is a 8000:

```
$ lt --port 8000
```

You will receive a url, for example `https://gqgh.localtunnel.me`, that you can share with anyone for as long as your local instance of localtunnel remains active. Any requests will be routed to your local service at the specified port.

Use received url anywhere in the development — specify in settings, set as domain (`/setdomain` command) for `@BotFather` and browse it - `https://gqgh.localtunnel.me/redirect/` instead of `127.0.0.1:8000/redirect/`.

1.1.2 Installation

Instal package via `pip`:

```
$ pip install django-telegram-login
```

1.1.3 Settings

Add application to the installed apps:

```
INSTALLED_APPS = [
    ...
    'django_telegram_login',
]
```

If you use only one bot you are able to add the following settings to own `settings.py` in Django project:

```
TELEGRAM_BOT_NAME = 'django_telegram_login_bot'
TELEGRAM_BOT_TOKEN = '459236585:AAEee0Ba4fRijf1BRNbBO9W-Ar15F2xgV98'
TELEGRAM_LOGIN_REDIRECT_URL = 'https://gqgh.localtunnel.me'
```

And use them in cases below:

```
from django.conf import settings


bot_name = settings.TELEGRAM_BOT_NAME
bot_token = settings.TELEGRAM_BOT_TOKEN
redirect_url = settings.TELEGRAM_LOGIN_REDIRECT_URL
```

But `django-telegram-login` allows you to use unlimited bots as you will see during learing the package.

1.2 How to use widgets

1.2.1 Interface

For now you are able to customize widget by the following arguments:

1. Size: small, medium, large.
2. User photo: show it near the button or not.
3. Corner radius: radius of corners from material to bootstrap style.

1.2.2 Login widgets types

You are able to make two reactions for user interaction with a button: `callback` and `redirect`. Telegram response with the following data relates to login widgtes types:

1. first_name: first name.
2. last_name: last name.
3. username: username.
4. photo_url: link to user's photo that located in Telegram storage
5. auth_date: Unix datetime (time in second from time when Unix was born)

6. hash: secret thing to verify data above.

Callback allows you to handle user data currently on page - you will receive data from Telegram in special `JavaScript-function-handler` (you need to implement it but save the name).

```
<script type="text/javascript">
    function onTelegramAuth(user) {
        alert('Logged in as ' + user.first_name + ' ' + user.last_name + '!');
    }
</script>
```

So you can handle it on the front-end or make a `AJAX` call to back-end and transfer a data.

Redirect transfers user to the specified link.

```
telegram_login_widget = create_redirect_login_widget(
    redirect_url, bot_name, size=LARGE, user_photo=DISABLE_USER_PHOTO
)
```

It will contain an user's data in get request parameters.

```
[12/Feb/2018 03:32:04] "GET /
    ?id=299661134
    &first_name=Dmytro
    &last_name=Striletskyi
    &username=dmytrostriletskyi
    &photo_url=https%3A%2F%2Ft.me%2Fi%2Fuserpic%2F320%2Fdmytrostriletskyi.jpg
    &auth_date=1518406180
    &hash=f5cd61a87131fcf51fc745d465a36bdcc58db4175ccac7c5afbf641359f55807
    HTTP/1.1" 200 14
```

So get it in `request.GET` within your view that handle request on specified URL.

1.2.3 Customizing

Customize **widgets interface** with the following parameters:

1. `Size`: constants `SMALL`, `MEDIUM`, `LARGE`.
2. `User photo`: do not pass any parameters to enable by defaul or disable with `DISABLE_USER_PHOTO` constant.
3. `Corner radius`: integer in range from 1 (material) to 20 (bootstrap, by default).

Customize **widgets login type** with the following parameters:

1. `Bot name`: name of bot as string.
2. `Redirect URL` (required only for the redirect widget): website address that will receive user's data by get request.

Import size constants, corner radius and a constant for disabling photo.

```
from django_telegram_login.widgets.constants import (
    SMALL,
    MEDIUM,
    LARGE,
    DISABLE_USER_PHOTO,
)
```

Import widget generators functions.

```
from django_telegram_login.widgets.generator import (
    create_callback_login_widget,
    create_redirect_login_widget,
)
```

Generate widgets according to provided functions.

```
telegram_callback_login_widget = create_callback_login_widget(bot_name, corner_
↪radius=10, size=SMALL)

telegram_callback_llogin_widget = create_redirect_login_widget(
    redirect_url, bot_name, size=LARGE, user_photo=DISABLE_USER_PHOTO
)
```

1.2.4 Rendering

Widget generator returns a string that contains `JavaScript` code. This code creates widget (button) automatically and handles user taps (requests) on its own. Your deal is to receive and process user data.

So use it in your views via context.

```
def callback(request):
    telegram_login_widget = create_callback_login_widget(bot_name, size=SMALL)

    context = {'telegram_login_widget': telegram_login_widget}
    return render(request, 'telegram_auth/callback.html', context)


def redirect(request):
    telegram_login_widget = create_redirect_login_widget(
        redirect_url, bot_name, size=LARGE, user_photo=DISABLE_USER_PHOTO
    )

    context = {'telegram_login_widget': telegram_login_widget}
    return render(request, 'telegram_auth/redirect.html', context)
```

Do not forget to make its rendering safe, because it is not a raw text but `Javascript`. Below is an example of a `Jinja code`.

```
{% autoescape off %} {{ telegram_login_widget }} {% endautoescape %}
```

1.2.5 Telegram authentication

There may be the situations, when hackers will send you incorrect Telegram data (pretending to be from a real user). `django-telegram-login` provides the following way to ensure that data is correct and isn't hacked.

```
from django_telegram_login.authentication import verify_telegram_authentication
from django_telegram_login.errors import (
    NotTelegramDataError,
    TelegramDataIsOutdatedError,
)
```

(continues on next page)

(continued from previous page)

```
def index(request):

    # Initially, the index page may have no get params in URL
    # For example, if it is a home page, a user should be redirected from the widget
    if not request.GET.get('hash'):
        return HttpResponse('Handle the missing Telegram data in the response.')

    try:
        result = verify_telegram_authentication(bot_token=bot_token, request_
↪data=request.GET)

    except TelegramDataIsOutdatedError:
        return HttpResponse('Authentication was received more than a day ago.')

    except NotTelegramDataError:
        return HttpResponse('The data is not related to Telegram!')

    # Or handle it as you wish. For instance, save to DB.
    return HttpResponse('Hello, ' + result['first_name'] + '!')
```

`verify_telegram_authentication` implements Telegram instructions to verify the authentication. If result does not raise errors, it will return a dictionary with user data.

Errors:

1. `NotTelegramDataError` - the verification algorithm did not authorize Telegram data.
2. `TelegramDataIsOutdatedError` - The Telegram data is outdated.

1.3 Contribution

1.3.1 Issues

If you find an issue or a bug, or you want to request any feautre, feel free to use Github issues.

1.3.2 Pull requests

If you want to impove `django-telegram-login` or resolve any issue, please use Github pull requests.

1.3.3 Development

Follow this code style in your development, please.

Install the packages required for development:

```
$ pip install -r requirements-dev.txt
```

Run the tests before development to make sure `django-telegram-login` logic works properly:

```
$ python -m unittest discover
```

Follow a codestyle with the following linters:

```
$ flake8 django_telegram_login && pycodestyle django_telegram_login && pylint django_
↪telegram_login
```