

---

# **django-private-chat Documentation**

***Release 0.2.2***

**delneg**

**Mar 31, 2019**



---

## Contents

---

<b>1</b>	<b>:sunglasses: django-private-chat :sunglasses:</b>	<b>3</b>
1.1	Important Notes . . . . .	3
1.2	Documentation . . . . .	4
1.3	Example project . . . . .	4
1.4	Customize the templates . . . . .	4
1.5	Exsiting project quickstart . . . . .	4
1.6	Running Tests . . . . .	6
1.7	Credits . . . . .	6
<b>2</b>	<b>Installation</b>	<b>7</b>
<b>3</b>	<b>Usage</b>	<b>9</b>
<b>4</b>	<b>Messages</b>	<b>11</b>
<b>5</b>	<b>Settings</b>	<b>13</b>
<b>6</b>	<b>Admin</b>	<b>15</b>
<b>7</b>	<b>Starting the server</b>	<b>17</b>
<b>8</b>	<b>Contributing</b>	<b>19</b>
8.1	Types of Contributions . . . . .	19
8.2	Get Started! . . . . .	20
8.3	Pull Request Guidelines . . . . .	21
8.4	Tips . . . . .	21
<b>9</b>	<b>Credits</b>	<b>23</b>
9.1	Development Lead . . . . .	23
9.2	Contributors . . . . .	23
<b>10</b>	<b>0.2.2 (2018-12-12)</b>	<b>25</b>
<b>11</b>	<b>0.2.1 (2018-12-07)</b>	<b>27</b>
<b>12</b>	<b>0.2.0 (2018-10-22)</b>	<b>29</b>
<b>13</b>	<b>0.1.9 (2018-07-16)</b>	<b>31</b>

<b>14</b>	<b>0.1.8 (2018-03-23)</b>	<b>33</b>
<b>15</b>	<b>0.1.7 (2018-03-20)</b>	<b>35</b>
<b>16</b>	<b>0.1.6 (2017-04-11)</b>	<b>37</b>
<b>17</b>	<b>0.1.5 (2017-03-11)</b>	<b>39</b>
<b>18</b>	<b>0.1.4 (2017-02-12)</b>	<b>41</b>
<b>19</b>	<b>0.1.3 (2017-02-11)</b>	<b>43</b>
<b>20</b>	<b>0.1.2 (2017-02-11)</b>	<b>45</b>
<b>21</b>	<b>0.1.1 (2017-02-10)</b>	<b>47</b>
<b>22</b>	<b>0.1.0 (2017-02-10)</b>	<b>49</b>

Contents:



# CHAPTER 1

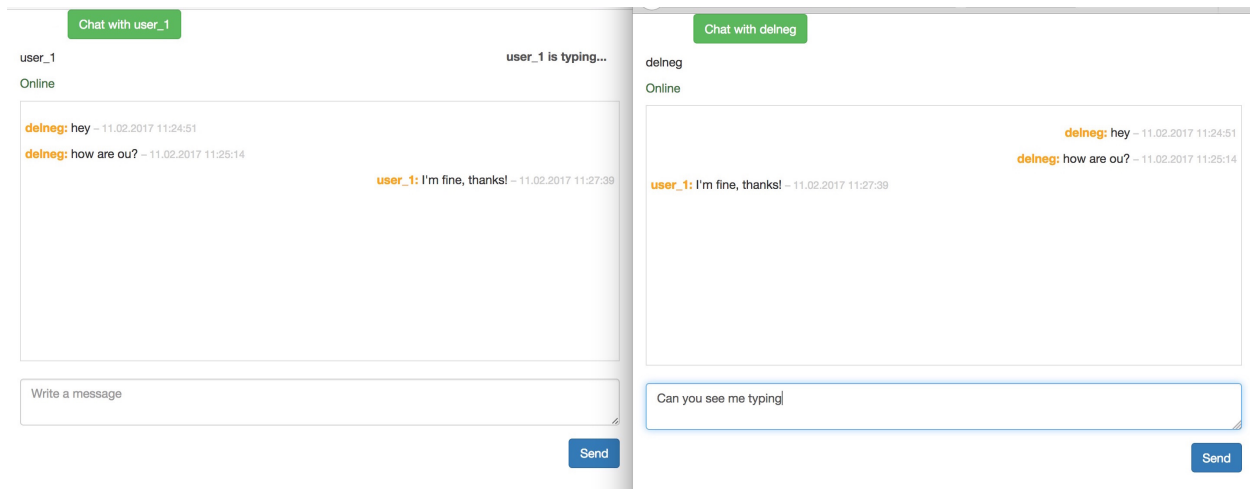
---

:sunglasses: django-private-chat :sunglasses:

---

Please also check out our another package [https://github.com/Bearle/django\\_mail\\_admin](https://github.com/Bearle/django_mail_admin)

Django one-to-one Websocket-based Asyncio-handled chat, developed by Bearle team



## 1.1 Important Notes

This app uses separate management command, `run_chat_server` for running Websockets in Django context. It is intended to be used with something like Supervisor or Systemd to run asyncio webserver as a separate one from Django. We didn't want our app to be limited to be used together with Django Channels - that's why we did it that way.

You can find an example Systemd config to run it as a service at <https://github.com/Bearle/django-private-chat/blob/dev/example.service>

P.S. Don't forget to change `CHAT_WS_SERVER_HOST` && `CHAT_WS_SERVER_PORT` && `CHAT_WS_SERVER_PROTOCOL` settings!

## 1.2 Documentation

The full documentation is (finally) at <https://django-private-chat.readthedocs.io> . You can also check the docstrings & this readme.

## 1.3 Example project

You can check out our example project by cloning the repo and heading into `example/` directory. There is a README file for you to check, initial data to check out the chat included.

## 1.4 Customize the templates

How to customize the template? Just copy:

```
venv/lib/pythonX.X/site-packages/django_private_chat/templates/django_private_chat/  
→dialogs.html  
to  
yourapp/templates/django_private_chat/dialogs.html
```

And feel free to edit it as you like! We intentionally left the JS code inside for it to be editable easily.

## 1.5 Exsiting project quickstart

Install django-private-chat:

```
pip install django-private-chat
```

Migrate:

```
python manage.py migrate django-private-chat
```

Note: you can use this package with or without uvloop, just run either

```
python manage.py run_chat_server
```

or run

```
python manage.py run_chat_server_uvloop
```

Add it to your *INSTALLED\_APPS*:

```
INSTALLED_APPS = (  
    ...  
    'django_private_chat',  
    ...  
)
```

Add the server & port for your asyncio server to settings:



```
CHAT_WS_SERVER_HOST = 'localhost'
CHAT_WS_SERVER_PORT = 5002
CHAT_WS_SERVER_PROTOCOL = 'ws'
```

It is possible to change messages datetime format using

```
DATETIME_FORMAT
```

Add django-private-chat's URL patterns:

```
from django_private_chat import urls as django_private_chat_urls

urlpatterns = [
    ...
    url(r'^$', include('django_private_chat.urls')),
    ...
]
```

Add

```
{% block extra_js %}{% endblock extra_js %}
```

to your base template

Now you can start a dialog using

```
/dialogs/some_existing_username
```

To create a WSS (TLS) server instead:

```
python manage.py run_chat_server "path/to/cert.pem"
```

(also works with uvloop). The “cert.pem” file should be a plaintext PEM file containing first a private key, then a certificate (may be a concatenation of a .key and a .crt file). Please note that wss will use TLSv1 by default for python 3.5 & 3.4 and will use ssl.PROTOCOL\_TLS\_SERVER for 3.6 and above. Features —

- :white\_check\_mark: Uses current app model (get\_user\_model()) and settings.AUTH\_USER\_MODEL)
- :white\_check\_mark: Translatable (uses ugettext and {% trans %} )
- :white\_check\_mark: One-to-one user chat
- :white\_check\_mark: Works using WebSockets
- :white\_check\_mark: Works (optionally) using WSS (TLS) connections (disclaimer - security not guaranteed)
- :white\_check\_mark: Displays online/offline status
- :white\_check\_mark: Display typing/not typing status
- :white\_check\_mark: Soft deletable message model - be sure to keep messages to comply with message-keeping laws
- :white\_check\_mark: Flash the dialog button when the user you are not currently talking to wrote you a message
- :point\_right: TODO: add a dialog to the list when new one started
- :point\_right: TODO: add user-not-found and other alerts
- :point\_right: possible Redis backend intergration

## 1.6 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

## 1.7 Credits

Tools used in rendering this package:

- [Cookiecutter](#)
- [cookiecutter-djangopackage](#)

## CHAPTER 2

---

### Installation

---

At the command line:

```
$ pip install django-private-chat
```



## CHAPTER 3

---

### Usage

---

To use django-private-chat in a project, add it to your *INSTALLED\_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_private_chat',
    ...
)
```

Add the server & port for your asyncio server to settings:

```
CHAT_WS_SERVER_HOST = 'localhost'
CHAT_WS_SERVER_PORT = 5002
CHAT_WS_SERVER_PROTOCOL = 'ws'
```

Add django-private-chat's URL patterns:

```
from django_private_chat import urls as django_private_chat_urls

urlpatterns = [
    ...
    url(r'^$', include(django_private_chat_urls)),
    ...
]
```

Add

```
{% block extra_js %}{% endblock extra_js %}
```

to your base template

Migrate:

```
python manage.py migrate django-private-chat
```

Now start the chat server:

```
python manage.py run_chat_server
```

## CHAPTER 4

---

### Messages

---

Application provides the following message channels:

```
'new-message',  
'new-user',  
'online',  
'offline',  
'check-online',  
'is-typing',  
'read_message'
```

which are pretty self-explanatory.

Here is detailed explanation of what each channel does and data types:

**new-message** Example from js:

```
{  
  type: 'new-message',  
  session_key: '{{ request.session.session_key }}',  
  username: opponent_username,  
  message: message  
}
```

In the handler, a new Message object is created and the received packet along with the additional parameters is sene to the other user's websocket (if present)

```
packet['created'] = msg.get_formatted_create_datetime()  
packet['sender_name'] = msg.sender.username  
packet['message_id'] = msg.id
```

**new-user** Sends connected client list of currently active users.

```
# Get list list of current active users  
users = [  
    {'username': username, 'uuid': uuid_str}
```

(continues on next page)

(continued from previous page)

```
    for username, uuid_str in ws_connections.values()
]

# Make packet with list of new users (sorted by username)
packet = {
    'type': 'users-changed',
    'value': sorted(users, key=lambda i: i['username'])
}
```

**online** Informs the users when someone of other has gone online.

```
{'type': 'gone-online', 'usernames': [user_owner.username]}
```

**offline** Distributes the users 'gone offline' status to everyone he has dialog with {'type': 'gone-offline', 'username': user\_owner.username}

**check-online** Same as online, except that it is used to provide the user that has gone online with information about who of his dialogs' users is online.

**is-typing** Shows message to opponent if the user is typing a message

```
{'type': 'opponent-typing', 'username': user_opponent}
```

**read\_message** Send message to user if the opponent has read the message Also sets the message.read to *True*.

```
{'type': 'opponent-read-message', 'username': user_opponent, 'message_id': message_id}
```



# CHAPTER 5

---

## Settings

---

You should specify settings in your settings.py like this:

```
CHAT_WS_SERVER_HOST = 'localhost'
CHAT_WS_SERVER_PORT = 5002
CHAT_WS_SERVER_PROTOCOL = 'ws'
DATETIME_FORMAT = "d.m.Y H:i:s"
```

Here's a list of available settings:

```
CHAT_WS_SERVER_PROTOCOL - 'ws' or 'wss'
CHAT_WS_SERVER_HOST - 'localhost' or ip or domain
CHAT_WS_SERVER_PORT - websocket application port
DATETIME_FORMAT - "d.m.Y H:i:s" - format for datetimes
```



Application provides django admin intergration for Dialog and Message models.

In order to provide custom admin representation, first you have to unregister existing:

```
from django_private_chat.models import Dialog, Message
admin.site.unregister(Dialog)
admin.site.unregister(Message)

// your example admin
class DialogAdmin(admin.ModelAdmin):
    list_display = ('id',)
admin.site.register(Dialog, DialogAdmin)
```



## CHAPTER 7

---

### Starting the server

---

Application provides two managements commands, `run_chat_server` and `run_chat_server_uvloop`.

That means that asyncio server is started **SEPARATELY** from the main Django application. You can also supply optional “path/to/cert.pem” to the command to use wss.

What management command do is they simply get the asyncio/uvloop event loop, add handlers for different message types to it and run the loop forever.



Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 8.1 Types of Contributions

### 8.1.1 Report Bugs

Report bugs at <https://github.com/Bearle/django-private-chat/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 8.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

### 8.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

### 8.1.4 Write Documentation

django-private-chat could always use more documentation, whether as part of the official django-private-chat docs, in docstrings, or even on the web in blog posts, articles, and such.

### 8.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/Bearle/django-private-chat/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 8.2 Get Started!

Ready to contribute? Here's how to set up *django-private-chat* for local development.

1. Fork the *django-private-chat* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-private-chat.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-private-chat
$ cd django-private-chat/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_private_chat tests
$ python setup.py test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.



## 8.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

## 8.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_private_chat
```



### 9.1 Development Lead

- delneg <tech@bearle.ru>
- guitarmustafa <oleg5432101@rambler.ru>

### 9.2 Contributors

None yet. Why not be the first?

History



## CHAPTER 10

---

0.2.2 (2018-12-12)

---

- Fix read\_message\_handler by idonoso



## CHAPTER 11

---

0.2.1 (2018-12-07)

---

- Compatibility with python3.7 by Emeka Icha





## CHAPTER 12

---

0.2.0 (2018-10-22)

---

- Added WSS fix for python 3.4 & 3.5



## CHAPTER 13

---

0.1.9 (2018-07-16)

---

- Added WSS support by @zsmith3



## CHAPTER 14

---

0.1.8 (2018-03-23)

---

- Fixed time in Message model to be timezone-aware



## CHAPTER 15

---

0.1.7 (2018-03-20)

---

- Additions for django 2.0





## CHAPTER 16

---

0.1.6 (2017-04-11)

---

- Fixed bugs with static files and added comment about extra\_js block to readme



## CHAPTER 17

---

0.1.5 (2017-03-11)

---

- Added flashing other user button when he sent you a message and you're in another dialog



## CHAPTER 18

---

0.1.4 (2017-02-12)

---

- Added support for django 1.8,1.9



## CHAPTER 19

---

0.1.3 (2017-02-11)

---

- Removed uvloop from requirements





## CHAPTER 20

---

0.1.2 (2017-02-11)

---

- Fixed i18n not loaded in dialogs template bug



## CHAPTER 21

---

0.1.1 (2017-02-10)

---

- Added migrations.



## CHAPTER 22

---

0.1.0 (2017-02-10)

---

- First release on PyPI.