django-intercom Documentation

Release 0.0.13

Ken Cochrane

Apr 21, 2017

Contents

Contents:

CHAPTER 1

Installation

1. Install django-intercom using pip:

```
pip install django-intercom
```

2. Add intercom to your INSTALLED_APPS in your django settings file:

```
INSTALLED_APPS = (
    # all
    # other
    # apps
    'intercom',
)
```

3. Add "INTERCOM_APPID" setting to your django settings file with your intercom application Id.

   in settings.py:

```
INTERCOM_APPID = "your appID"
```

CHAPTER 2

Usage

Add the template tag code into your base template before the body tag.

At the top of the page put this:

```
{% load intercom %}
```

At the bottom of the page before the </body> tag put this:

```
{% intercom_tag %}
```

CHAPTER 3

Enable Secure Mode

This is optional, if it isn't set, then you will not use secure mode.

If you want to turn on secure mode, you can add INTERCOM_SECURE_KEY to your settings.py with the private key you can get from your intercom->app->security page.

in settings.py:

```
INTERCOM_SECURE_KEY = "your security_code"
```

You will need to look in the code samples to find the security key.

You will also need to make sure you check the "Enable secure mode" check box on the security page before this will work correctly.

CHAPTER 4

Intercom Inbox

Intercom has the ability to add an inbox link to your app so that people can contact you, and for you to let them know when they have a message waiting. If you would like to use these features you need to do the following.

1. Add the intercom css id to any inline element containing text, for example:

```
<a id="Intercom" href="#">Support</a>
```

2. Add the appropriate CSS to your style sheet.

No Icon:

```
#Intercom {
  display: inline-block;
  text-decoration: underline;
  padding: 0;
}
```

White Envelope (white text on black background):

```
#Intercom {
  display: inline-block;
  text-decoration: underline;
  padding: 0 0 0 24px;
  background: transparent url(https://www.intercom.io/images/white_env.png) no-repeat␣
↪left center;
}
```

Black Envelope (black text on white/grey background):

```
#Intercom {
  display: inline-block;
  text-decoration: underline;
  padding: 0 0 0 24px;
  background: transparent url(https://www.intercom.io/images/black_env.png) no-repeat␣
↪left center;
}
```

If you want to show the unread message count then also add the following:

```
#Intercom em {
  display: inline-block;
  font-style: normal;
  text-decoration: underline;
}
```

3. Configure your settings. Add the following to your django settings if you would like to change the defaults.

INTERCOM_ENABLE_INBOX

Default: True

In settings.py:

```
INTERCOM_ENABLE_INBOX = True
```

INTERCOM_ENABLE_INBOX_COUNTER

Default: True

In settings.py:

```
INTERCOM_ENABLE_INBOX_COUNTER = True
```

INTERCOM_INBOX_CSS_SELECTOR

Default: '#Intercom'

In settings.py:

```
INTERCOM_INBOX_CSS_SELECTOR = '#Intercom'
```

CHAPTER 5

Custom Data

Intercom.io allows you to send them your own custom data, django-intercom makes this easy. All you need to do it create a Class with a custom_data method that accepts a Django User model as an argument and returns a dictionary. Here is an example:

```
from thepostman.models import message

class IntercomCustomData:
    """ Custom data class located anywhere in your project
        This one is located in thepostman/utils/custom_data.py """

    def custom_data(self, user):
        """ Required method, same name and only accepts one attribute (django User␣
↪model) """

        num_messages = message.objects.filter(user=user).count()
        num_unread = messages.objects.filter(user=user, read=False).count()

        return {
            'num_messages' : num_messages,
            'num_unread' : num_unread,
        }
```

Once you have your classes built, you will need to register them with django-intercom so that it knows where to find them. You do this by adding the class to the INTERCOM_CUSTOM_DATA_CLASSES setting. It is important to note that if you have the same dict key returned in more then one Custom Data Class the last class that is run (lower in the list) will overwrite the previous ones.

INTERCOM_CUSTOM_DATA_CLASSES

Default = None

in settings.py:

```
INTERCOM_CUSTOM_DATA_CLASSES = [
    'thepostman.utils.custom_data.IntercomCustomData',
]
```

CHAPTER 6

Settings

These are the settings that you can add to your django settings file to control django-intercom.

INTERCOM_APPID

Required

Default: None

Example:

```
INTERCOM_APPID = "your appID"
```

INTERCOM_SECURE_KEY

Optional

Default: None

example:

```
INTERCOM_SECURE_KEY = "your security_code"
```

INTERCOM_INCLUDE_USERID

Optional

Whether to include the `user_id` in the intercom settings. Turn this off if you want to use the email as the key to identify a user.

Default: True

example:

```
INTERCOM_INCLUDE_USERID = False
```

INTERCOM_ENABLE_INBOX

Optional

Default: True

example:

```
INTERCOM_ENABLE_INBOX = True
```

INTERCOM_ENABLE_INBOX_COUNTER

Optional

Default: True

example:

```
INTERCOM_ENABLE_INBOX_COUNTER = True
```

INTERCOM_INBOX_CSS_SELECTOR

Optional

Default: '#Intercom'

example:

```
INTERCOM_INBOX_CSS_SELECTOR = '#Intercom'
```

INTERCOM_CUSTOM_DATA_CLASSES

Optional

Default = None

example:

```
INTERCOM_CUSTOM_DATA_CLASSES = [
    'thepostman.utils.custom_data.IntercomCustomData',
]
```

CHAPTER 7

People who have worked on this project

- Ken Cochrane (@KenCochrane)

CHAPTER 8

TODO

- Adding unit tests

CHAPTER 9

Indices and tables

- genindex
- search