django-intercom-io Documentation

Release 1.0.0

August 16, 2016

1 Development **3**

2 Contents **5**

django-intercom-io makes it easy for you to add support for **intercom.io** to your Django website.

CHAPTER 1

Development

The source repository can be found at https://github.com/eldarion/django-intercom-io

CHAPTER 2

Contents

2.1 ChangeLog

2.1.1 0.4

- change has function to use user.pk instead of email address

2.1.2 0.3

- add user id

2.1.3 0.2

- pass user to context for rendering template

2.1.4 0.1

- initial release

2.2 Installation

- To install

```
pip install django-intercom-io
```

- Add "intercom" to your INSTALLED_APPS setting:

```
INSTALLED_APPS = [
    # other apps
    "intercom",
]
```

2.3 Usage

At the top of your base template add:

```
{% load intercom_tags %}
```

And just before the `</body>` tag add:

```
{% intercom_js user %}
```

And if you want a feedback/support link, put:

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

somewhere (e.g. in your nav bar) as explained by the **intercom.io** documentation.

In your settings file set `INTERCOM_APP_ID` and optionally (if you use a user hash for security) `INTERCOM_USER_HASH_KEY` with the values provided by **intercom.io**.

2.3.1 Custom Data

intercom.io lets you send custom, per-user data to its site. **django-intercom** lets individual apps contribute what custom data they want to provide.

If you have an `INTERCOM_APPS` setting, it should be a list of apps that have an `intercom` module in them containing a `custom_data` function. This function should take a `user` and return a dictionary to be sent to **intercom.io** as custom data.

For example, if you have an app called `foo` with a `Foo` model, you might add:

```
INTERCOM_APPS = [
    "foo",
]
```

to your settings and then in `foo/intercom.py` have:

```
from foo.models import Foo

def custom_data(user):
    return {
        "foo_count" : Foo.objects.filter(user=user).count(),
    }
```