django-instant Documentation

Release 0.1

synw

Aug 30, 2017

Install

Instant delivers events in real time to the userland using the Centrifugo websockets server.

CHAPTER 1

Install and configure

1. Install Django Instant:

```
pip install django-instant
```

Add to installed apps:

```
"instant",
```

Set the urls:

```
# required only if you want to use the frontend at /instant/
url('^instant/', include('instant.urls')),
```

2. Install the websockets server

An installer is available for Linux only:

```
python3 manage.py installws
```

This will download the Centrifugo websockets server, install it and update your settings.py file with the appropriate configuration.

For other systems you have to install Centrifugo manually

Templates

Include the template `{% include "instant/client.html" %}` anywhere: nothing will be displayed it is the engine. See next section for messages handling.

Run the websockets server

```
python3 manage.py runws
```

Settings

Note: if you use the management command to install the server the settings will already be configured

Configure settings.py:

```
# required settings
CENTRIFUGO_SECRET_KEY = "70b651f6-775a-4949-982b-b387b31c1d84" # the_key_that_is_in_
↪config.json
SITE_SLUG = "my_site" # used internaly to prefix the channels
SITE_NAME = "My site"

# optionnal settings
CENTRIFUGO_HOST = 'http://ip_here' #default: localhost
CENTRIFUGO_PORT = 8012 # default: 8001
INSTANT_PUBLIC_CHANNEL = "public" #default: SITE_SLUG+'_public'
INSTANT_ENABLE_PUBLIC_CHANNEL = False # this one is to disable the default public␣
↪channel
```

By default the events are published using python. A go module is available to perform the publish operations in order to leave the main process alone as much as possible. This might be usefull when lots of messages are sent.

Note: when this option is enabled there is no error handling. This option is recommended when you need higher performance and don't care about error messages.

You might have to make the *instant/go/publish* file executable with *chmod*

```
INSTANT_BROADCAST_WITH = 'go'
```

Performance test: 1000 messages:

- Python: 2.96 seconds
- Go: 1.41 seconds

10000 messages:

- Python: 29.57 seconds
- Go: 14.44 seconds

Note: this test uses the standard publish function so that each new event sent makes an new connection to Centrifugo.

CHAPTER 2

Create channels

New in 0.6 to come (still in master for now)

Declare channels

To declare new channels in settings.py use this format:

```
( ("channel_name"), ("/path/where/to/connect/it",) )
```

If the second element of the tuple is set the channels will be connected only for the listed path. If not set the channel will autoconnect on every path. Example:

```
INSTANT_SUPERUSER_CHANNELS = (
 ("$mysite_admin1", ("/a/path", "/another/path")),
 ('$mysite_admin2',)
)
INSTANT_STAFF_CHANNELS = (
 ("$mysite_staff1", ("/a/path",)),
 ('$mysite_staff2',)
)
INSTANT_USERS_CHANNELS = (
 ('$mysite_users1',)
)
INSTANT_PUBLIC_CHANNELS = (
 ('mysite_public1',),
 ('mysite_public2',)
)
```

Handlers

To declare channels your must create a `/templates/instant/handlers/default.js` file that will be the default handler for the channels.

Each channel can have its own handler: just create it in the handlers directory: example: if a `$mychan` is declared in settings it is possible to create a `$mychan.js` file in the handlers directory to manage the handling logics just for that channel

CHAPTER 3

Send messages

There are two ways to publish messages:

Direct publish of messages

Go to `/instant/` as superuser and use the form to publish a message to a channel.

The next section will describe how to customize the handlers on the client side according to the event class.

Stream messages from code

```
from instant.producers import publish

# fire a message on the public channel whith error handling
err = publish(message='Hello world', event_class="infos", channel="public", data={
↪"myfield":"my_value"})
if err != None:
   print("Error", str(err))

# send an instant debug message during development
publish("Something happened somewhere in the code", event_class='debug')
```

The only required parameter is `message`.

To send extra data pass some json into `data`.

Note: if no channel is provided the events are published to the default public channel.

CHAPTER 4

Client-side event handlers

Messages handlers

You can customize the javascript that will handle each class of event of the default public channel.

Create a template `instant/extra_handlers.js` with this content:

```
{% include "mymodule/handlers.js" %}
```

In `templates/mymodule/handlers.js` define your client side handler:

```
if ( event_class == 'anyeventclass' ) {
        console.log(message);
}
```

Note: for javascript debugging you can set a `INSTANT_DEBUG = True` in settings.py

Note: this is only available for the default public channel events. To handle events in private channels see next section.

Connection handler

To perform custom actions on connect and disconnect events create templates:

Template *instant/events/connect.js* or *instant/events/disconnect.js*:

```
{% include "myapp/myjs.js %}
```

This javascript will be executed on the selected event

Message label formating

The default behavior pops up a message with a label according to the event_class. To change these default labels use the file `templates/instant/event_class_format.js`.

The default css classes are defined in `instant/static/instant.css`:

```
{
'default' : 'mq-label mq-default',
'important' : 'mq-label mq-important',
'ok' : 'mq-label mq-ok',
'info' : 'mq-label mq-info',
'debug' : 'mq-label mq-debug',
'warning' : 'mq-label mq-warning',
'error' : 'mq-label mq-error',
'object created' : 'mq-label mq-created',
'object edited' : 'mq-label mq-edited',
'object deleted' : 'mq-label mq-deleted',
}
```

Default icons (using font-awesome):

```
{
'default' : '<i class="fa fa-flash"></i>',
'important' : '<i class="fa fa-exclamation"></i>',
'ok' : '<i class="fa fa-thumbs-up"></i>',
'info' : '<i class="fa fa-info-circle"></i>',
'debug' : '<i class="fa fa-cog"></i>',
'warning' : '<i class="fa fa-exclamation"></i>',
'error' : '<i class="fa fa-exclamation-triangle"></i>',
'object edited' : '<i class="fa fa-pencil"></i>',
'object created' : '<i class="fa fa-plus"></i>',
'object deleted' : '<i class="fa fa-remove"></i>',
}
```

CHAPTER 5

Private channels

Add this in you main url file:

```
from instant.views import instant_auth

urlpatterns = [
     # ...
     url(r'^centrifuge/auth/$', instant_auth, name='instant-auth'),
     ]
```

All the channels prefixed with a dollar sign **$** are considered private.

```
from instant.producers import publish

publish(message='Private event', channel="$private_chan")
```

Security

All the messages sent to the Centrifugo server are signed using the secret key. When a client requests a connection to a private channels Centrifugo sends an ajax request to `/centrifuge/auth/` and expects to receive a signed response that will indicate if the user is authorized or not.

More info here about Centrifugo's auth mechanism.

Default channels

By default only the public channel is enabled. You can activate the default privates channels in settings:

```
# reserved to logged in users
INSTANT_ENABLE_USERS_CHANNEL = True
# reserved to staff users
```

```
INSTANT_ENABLE_STAFF_CHANNEL = True
# reserved to superuser
INSTANT_ENABLE_SUPERUSER_CHANNEL = True
```

Make sure the main client is loaded somewhere: `{% include "instant/client.html" %}` and add some handlers.

To push an event to one of theses channel use the `target` parameter: ‘

```
publish(message="Staff event", target="staff")
```

Note: if a `channel` parameter is provided, the `target` will be ignored.

To change the events behavior client-side customize the handlers like described in the previous section, and use the channel parameter:

```
if ( channel == "$mysite_staff" && event_class == 'anyeventclass' ) {
   console.log(message);
   return false
}
```

You can also overide the whole client: for the staff channel: `instant/channels/staff/js/client.js`

CHAPTER 6

Custom channels

Set you channels credentials in settings.py:

```
INSTANT_USERS_CHANNELS = ['$mychannel1']
INSTANT_STAFF_CHANNELS = ['$mychannel2']
INSTANT_SUPERUSER_CHANNELS = ['$mychannel3']
```

Now you can setup the client-side handlers for your channels:

Create a `{% instant/extra_clients.js %}` template that contains something like:

```
{% if user.is_authenticated and request.path == "/private/" %}
     {% include "myapp/client.js" %}
{% endif %}
```

Edit `myapp/client.js`:

```
var my_callbacks = {
    "message": function(dataset) {
         // the instantDebug variable is set via INSTANT_DEBUG = True in settings.py
     if (instantDebug === true) { console.log('EVENT: '+JSON.stringify(dataset));};
     res = unpack_data(dataset);
     var message = res['message']
     var event_class = res['event_class']
     var message_label = res['message_label']
     var data = res['data']
     var channel = res['channel'];
     if ( data.hasOwnProperty('my_field') ) {
                my_field = data['myfield']
     }
     // do something with the data
     console.log(message);
 },
     {% include "instant/js/join_events.js" %}
}
```

```
var subscription = centrifuge.subscribe("$mychannel", my_callbacks);
```

Custom auth function

You can write a custom auth backend to authenticate the user. Example: urls.py:

```
from mymodule.views import mychan_auth_view

url(r'^centrifuge/auth/$', mychan_auth_view),
```

In views.py:

```
import json
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.http.response import Http404
from instant.utils import signed_response

@csrf_exempt
def mychan_auth_view(request):
    if not request.is_ajax() or not request.method == "POST":
        raise Http404
    data = json.loads(request.body)
    channels = data["channels"]
    client = data['client']
    response = {}
    for channel in channels:
        response[channel] = {"status","403"}
        if channel == "$mychannel":
             # checks come here
             if request.user.is_authenticated() and whatever():
                response[channel] = signed_response(channel, client)
    return JsonResponse(response)
```