Trigger Happy - Goal

- Goal :

  Exchanging data from services arround the web

  To reach that : 3 tasks

  - Getting a token from a service, to be able to talk to it with its API
  - Grabbing the data from service A
  - Publishing the data to service B

Trigger Happy - ServiceMgr

ServicesMgr class

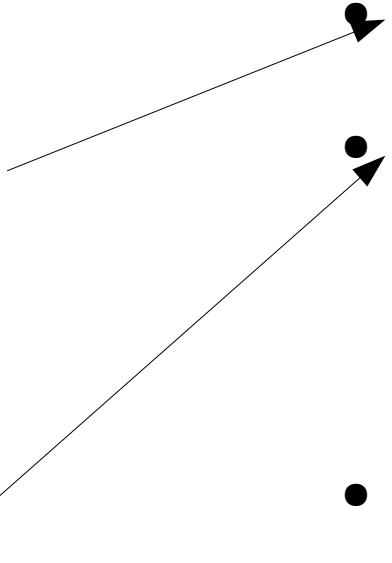
- read_data
- process_data
- save_data
- auth
- set_content

Helpers

- PublishingLimit
- HtmlEntities

Service Provider

- load_services



Trigger Happy - ServiceXXX

- ServiceXXX

  Each ServiceXXX inherits from Services (where XXX could be Twitter, GitHub, Mastodon and so on)
- That concerns Models (the Services Models), and Class (ServiceMgr class)

  The Model
- We add the field we want to be used to know what kind of data we want to grab or publish.

  For example for twitter, « screen » allow to know from which twitter account we want to get the data, or, on which twitter account we want to publish data

  The Class ServiceXXX
- Its token
- Its consumer key/id
- Does it use oauth1 or 2 or none

  The Methods ServiceXXX
- read_data : grabs data and push them in the cache
- save_data : stores/publish the data, getting from the cache, with the an API call by providing the token

  The Form are not inherits but the ProviderForm and ConsumerForm are almost always the same, this is why a XXXForm is inherit from XXXProviderForm and XXXConsumerForm

Trigger Happy - Helpers

PublishingLimit

- Allow the engine to detect if a trigger works fine, and if not, stop it, to avoid to disturb the other triggers. And specific message is display below the trigger

HtmlEntities

- To decode HTML entities to avoid weird render

Service Provider

- Load dynamically the services defined in settings.TH_SERVICES

Trigger Happy - Commands

Management Commands

- Trigger the retrieval of the data, by using

  python manage.py read

- Trigger the publishing of the data, by using

  python manage.py publish

- Recycling the data, by using

  python manage.py recycle

  a word about this last one.

- Often, services fail to handle requests and return error, such as « RateLimit » for example. When such a behavior is met, the data we wanted to published are refused by the service, but are not lost, as there are pushed in cache again, to be able to be pusblished at the next loop.