iwant_bot Documentation

Release

Kiwi.com interns

Aug 05, 2017

Contents:

CHAPTER 1

Top-level documentation

The slackbot accepts requests from users that want to do something and that would like not to be alone when doing them.

For example, Alice can tell the bot that she would like to have a coffee with somebody from 10:00, but she needs to know whether anyone is going to join her at 9:30 latest (otherwise she might decide not to have a coffee at all). Then, if Bob tells the bot at 9:20 that he would like to have a coffee during the next hour, the bot should put the two requests together. This means telling Bob that he joins Alice at 10:00 and telling Alice that she can have coffee with Bob.

Terminology

Request is defined by

- user who placed it,
- activity that is being requested,
- deadline — time the bot has to pair the user (`datetime.datetime`).
- activity start — the time in which the activity is supposed to start (`datetime.datetime`).
- activity duration — How long is the activity supposed to last (`float` — seconds).

Request resolution

1. The actual request is received via Slack (or other means).
2. It is parsed by the Slack backend and passed into request preprocessing pipeline.
3. In the pipeline, the request is
    - given an ID,
    - saved to a storage (typically to the central database), and

    - task is dispatched for the worker.
4. The storage assigns each request a provisional result.

Then comes the task worker:

1. The worker pulls stuff from the tasks queue.
2. According to what tasks arrive, it queries the storage for requests.
3. If a set of requests is resolved (e.g. a group of people can go and have a coffee together), it tells the storage that requests of those IDs are resolved.

Representation

Request

Requests are represented by a data structure, which is quite intuitive.

Notable facts:

- Each stored request is assigned a (degenerate) result.
- If the request is not matched, the result is in the `PENDING` state, so the matching failiure is obvious.
- If requests are matched, all start point to one result. The exact result that prevails is determined by the `iwant_bot.storage.RequestsStorage._find_fitting_result_id()` method. Results that become redundant by this act are invalidated.

Result

Results are represented by a data structure, which is also quite intuitive.

Notable facts:

- Results storage is denormalized:
  - The `iwant_bot.requests.Result.deadline` can be retrieved by querying requests in storage, but it is stored as a result property instead.
  - Aspects of the `iwant_bot.requests.Result.status` (pending, invalid, ???) status can be inferred by querying requests in the storage too.

CHAPTER 2

IWantBot packages

bot_worker package

Submodules

bot_worker.celery module

bot_worker.tasks module

Module contents

iwant_bot package

Submodules

iwant_bot.RequestCreator module

iwant_bot.ignore module

iwant_bot.iwant_process module

iwant_bot.pipeline module

iwant_bot.pool module

iwant_bot.request_acquisition module

iwant_bot.requests module

iwant_bot.slack_communicator module

iwant_bot.start module

iwant_bot.storage module

CHAPTER 3

Indices and tables

- genindex
- modindex
- search