Overview

flitter-agenda provides a wrapper for the Agenda scheduler. This allows jobs to be defined and scheduled programmatically, and persistently. It uses the same MongoDB database as the rest of the Flitter application.

For the most part, using flitter-agenda is identical to using vanilla Agenda. The only main difference is that jobs should be defined in individual files in a given directory (by default, `app/jobs`). Each of these files should export a class that extends `flitter-agenda/Job`. The name of a job is parsed from its file name, recursively. flitter-agenda uses the Flitter convention for sub-directory naming, meaning that sub-directories are delineated with a `:` character. For example, the job named `auth:CleanUsers` would be located in the file `app/jobs/auth/CleanUsers.job.js`.

Installation

flitter-agenda doesn't ship with Flitter by default, but it's pretty easy to add. First, install the package:

```
yarn add flitter-agenda
```

Now, modify the `Units.flitter.js` file. Add the add the following line to the "Custom Flitter Units" section to tell Flitter to load flitter-agenda:

```
  'Agenda': new (require('flitter-agenda/AgendaUnit'))(),
```

Et voilà! You should now have access to flitter-agenda.

Creating a Sample Job

As an example, we'll create a sample job that prints a some text to the console.

First, create a new job definition file. We can do this using the built in template with the `./flitter` command:

```
./flitter new job Log
```

This will create the file `app/jobs/Log.job.js`. Open it up, and define the actual logic for the job:

```
const Job = require('flitter-agenda/Job')
class LogJob extends Job {
    exec(job, done){

        const {msg} = job.attrs.data
        log(msg)

        done()
    }
}
module.exports = exports = LogJob
```

Here, we take the job argument `msg` and pass it to Flitter's `log()` function.

We can test out our job by running it from the Flitter shell. Here's a sample output:

```
sh-4.4$ ./flitter shell
powered by Flitter, © 2019 Garrett Mills(flitter)> _flit.sched.scheduler.schedule('in 3
seconds', 'Log', {msg: "This is a message!"})
Promise { <pending> }(flitter)> This is a message!
```

`_flit.sched.scheduler` is the running instance of Agenda, so you can call any function on it like you normally would such as `schedule()`, `every()`, even `define()`. (See the Agenda docs.) Here, we scheduled the `Log` job we created to run 3 seconds after the `schedule()` function is called, and passed it a message.

Revision #2
Created Tue, Mar 19, 2019 1:51 PM by Garrett Mills
Updated Tue, Mar 19, 2019 3:45 PM by Garrett Mills