Tometo

Jan 22, 2020

Contents:

CHAPTER 1

Installation

Tometo is functionally split into two parts — the frontend, which is a Vue.js app that's kept in the `./ui`, and the backend called Aph, which is an Elixir app that's kept in the `./aph` folder.

1.1 Prerequisites

In order to get the system running on your computer, you'll need some prerequisites:

- Git
- A PostgreSQL server, and its development headers (sometimes called `libpq-dev` or `postgresql-devel`)
- Elixir (and Erlang, but usually those two are installed side-by-side)
- Node.js, the latest LTS or Stable version should work
- Python 3 and `pip` (plus development headers, sometimes separate as `python3-devel`)
- eSpeak (and its development headers, sometimes separate as `espeak-devel`)
- FFmpeg

Optionally, there's some more advanced features you can enable, which need some of these dependencies:

- A Google Cloud service account API key that has access to the Text-To-Speech API

Note: These still need to be documented properly.

1.2 Automatic Installation

Before doing any installation, make sure you have Elixir's package manager scripts downloaded locally! Run this command to do so:

```
mix local.hex
```

If you have all of the prerequisites installed, you can try cloning the repository and running our automatic setup script:

```
git clone ssh://vcs@marisa.cloud:2222/source/tometo.git
# or with HTTPS
git clone https://marisa.cloud/source/tometo.git
cd tometo/
script/localsetup
```

This will set up everything for you, the only thing you need to do yourself is fill in the config file for Aph and run the database setup. The config file can be found in *aph/config/config.exs*. This is where you fill in your database credentials. After you're done doing that, you can then run the database setup:

```
cd aph
mix ecto.create
```

1.3 Manual Installation

You'll want to install `aeneas`, which parses text for us (this needs ffmpeg and espeak installed and available):

```
pip3 install --user numpy
pip3 install --user aeneas
```

Check that it's installed correctly:

```
python3 -m aeneas.diagnostics
```

Then, you can clone the repository.

```
git clone https://marisa.cloud/source/tometo.git
```

Once you're in the directoy, you'll want to install the dependencies for the frontend:

```
npm install
```

And the backend:

```
cd aph
mix deps.get
```

Now you can go ahead and copy the backend configuration file:

```
cp aph/config/config.example.exs aph/config/config.exs
```

Next, to create the necessary database tables and configuration, fill in your database configuration in `aph/config/config.exs` and run this:

```
cd aph
mix ecto.create
```

As a final step, you should copy the example config files:

```
cp .env.example .env
```

1.4 Configuration

Configuration is (unfortunately) different for frontend and backend. The frontend loads environment variables either through you directly setting them or through `.env`, while Aph loads its own config contained in `aph/config/`.

Note: TODO: Add production configuration info here

1.5 Running

We have multiple scripts to provide some common uses if you're planning on working on Tometo. These include:

- `script/build`: Runs a production build
- `script/lint`: Makes sure your code looks nice and is ready to commit
- `script/run`: Runs both the frontend and the backend
- `script/run_b`: Runs only the backend
- `script/run_f`: Runs only the frontend
- `script/watch`: Runs and watches for changes for the frontend and backend. This is what you want most of the time.

CHAPTER 2

API Details

Tometo's API, called Aph, takes care of everything in the backend, such as interfacing with the database, generating statuses, and so forth. We have a couple of rules when working with Aph that we try to follow in order to make a consistent effort at keeping the codebase clean and understandable.

2.1 API Interface

We follow the RESTful API standards. This means a couple of things:

- Use proper HTTP statuses (i.e. `201` on a successful `POST`)
- Routes should be: - Plural unless referring to a singleton object (`/api/statuses`, `/api/users`)
  - Usable by multiple methods (`GET /api/users`, `POST /api/users`
  - Used for specific cases only when it's unavoidable (`/api/users/:id/invitations`)
- When returning an error, put it in this format:

```
{
  "error": true,
  "id": "unique-error-id",
  "message": "Helpful error message"
}
```

- Always return full resource objects, not only subsections of them. For example, when creating a user, don't only return the username, return the entire thing, even if it might seem superfluous.

2.2 General Elixir/Phoenix recommendations

We don't use a `data` key in views, like the Phoenix generators would like you to believe. Most things in our JSON returns are top-level.

`AphWeb.ErrorView` contains the most common error codes. Some of them accept custom messages, some don't — feel free to modify one to include a message, just be sure to check where else the error is being used from first!

When you want to make sure that only logged in users can access a route, do this:

```
use AphWeb.Authorize

plug :user_check when action in [:your_action_name]
```

Always use brackets around method calls, it's cleaner and easier to understand. Piping is usually not worth it when it's only a single pipe (unless you're working with a `conn` struct).

Linting can be done via `script/lint`, or by explicitly cd-ing into the aph/ subdirectory and running `mix format` there.

CHAPTER 3

Contributing to Tometo

Thanks for your interest in contributing to Tometo! There is more than one way to contribute to the project, and we appreciate all of them.

If you have any questions, feel free to ask them in the Tometo Discord Server. As a reminder, all contributors, in whichever way, are required to follow the Code of Conduct.

3.1 Feature Requests

If you have ideas or even concrete suggestions for a feature or an enhancement, it's best to discuss it first on Discord before diving into the specifics. If you feel up to implement it, open a *pull request*!

3.2 Bug Reports

While we wish everything in here worked perfectly, bugs are natural to occur. This is why we recommend that, even when you're not sure it's a bug, to report it anyways.

Please search for already existing or similar reports first before you open an issue.

It also makes our lives much easier if the issue title is as descriptive as possible. This can be the specific error message given, the steps you used to reproduce it, or something else that's unique to the bug you're experiencing.

Filing a bug report happens on our bug tracker, and you can find a link to create a new issue here.

If you're running Tometo locally, please include as much information as possible about your setup, what Rust version you use, what Node version, your operating system, stack traces, et cetera. To get nice backtraces from Rust (e.g. Otemot), there's a special environment variable you can set:

```
RUST_BACKTRACE=1 cargo run
```

3.3 Pull Reqests

Pull Requests are how we usually land code for Tometo. We use a simple model, where every contributor pushes changes either to their fork or to a branch on the main repository, and then brings those changes into the master branch.

Always make sure that your code conforms to the style guidelines by running the following:

```
script/lint
```

Please don't annoy anyone to review your pull request, people have limited time, and sometimes other things have priorities.

3.4 Issue Triage

Issues on the bug tracker can be for different *trackers*, and have other configuration options. Here's some notes on them:

- Don't set a priority unless you know what you're doing
- The `Epic` tracker refers to issues that track a multitude of other issues, e.g. for a large feature

CHAPTER 4

Releasing Tometo

Releasing software can be an arduous task, which is why we're trying really hard to make it as simple as possible for Tometo. Here's a bunch of information and guides you might appreciate if you're a person who is in charge of releases.

4.1 Basics

Right now, a Tometo release is basically just:

- A tag in the source repository (something like `0.4.2`)
- A corresponding post on the Tometo Release Blog
- For larger releases (such as *0.x* or *x.0*), a post on the main Tometo website at https://tometo.org

4.1.1 Preparing a Release

If you have commit access, you can make one commit that includes the version number changes. That's basically all that's required code-wise. Change the version numbers in these places:

- `package.json`
- `aph/mix.exs`

Call the commit the version number, and create a tag that refers to that commit:

```
git tag x.y.z
```

Then push them:

```
git push origin master
git push origin master --tags
```

4.1.2 Writing a release post

Every release, even minor ones, should have a post on the Release Blog. Assuming you have correct permissions to create blog posts there, the minimum a blog post should contain is a changelog that lists all features, bug fixes and enhancements, ESPECIALLY the ones that were made by people outside of the usual committers. If there is very menial stuff authored by usual committers, such as changing some links in documentation or something, feel free to leave those out, but don't make assumptions.

That being said, it's your blog post! You can put as many fun little things into it as you'd like. Once you're done, let the team know and we'll push the release out onto our social media channels and the Discord. Congrats!!

CHAPTER 5

Code of Conduct

5.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.3 Our Responsibilities

The Tometo team is responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

The Tometo team has the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

5.4 Scope

This Code of Conduct applies within all project spaces, and it also applies when an individual is representing the project or its community in public spaces. This applies to the project GitLab instance, the Discord server, but also the official Twitter account. Representation of a project may be further defined and clarified by project maintainers.

5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project lead at marisa@mokou.zone. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The Tometo team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

5.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

For answers to common questions about this code of conduct, see https://www.contributor-covenant.org/faq

This is the central documentation for the Tometo project.