Know Me API Documentation

Release 1.5.0

Chathan Driehuys

Jan 21, 2022

Contents:

CHAPTER 1

Know Me API

API Root https://toolbox.knowmetools.com

API Endpoint Documentation https://toolbox.knowmetools.com/docs/

Developer Documentation http://know-me-api.readthedocs.io/

1.1 About

This is the API behind our Know Me app. It is written in Python using Django and Django Rest Framework.

1.1.1 Development

The API is built with Python using Django and Django Rest Framework.

Recommended Development Environment

If you are comfortable with setting up a python development environment and cloning the project, feel free to skip to the *development environment overview*.

Prerequisites

We use git for version control and generally follow the development model laid out here. If you are looking for a tool to assist in following this model, we recommend git-flow, a tool made by the same people that created the development model.

This project runs on Python 3.6. You must have Python 3.6 installed to contribute. If you do not have it installed, you can find it here. To manage third-party Python packages, we use pipenv.

Project Setup

The first step is to clone the environment:

```
$ git clone git@github.com:knowmetools/km-api
# Or, if you don't have SSH access
$ git clone https://github.com/knowmetools/km-api

# The remaining commands require you to be in the project directory
$ cd km-api/
```

If you installed the `git-flow` extension, you can now setup the repository to use it:

```
$ git flow init -d
```

We also recommend the following configuration options:

```
$ git config gitflow.feature.finish.keepremote true
$ git config gitflow.bugfix.finish.keepremote true
```

Finally, install the project requirements

```
$ pipenv install --dev
```

The final step is to create a `.env` file in the root of your project. This file will be read by any commands executed through `pipenv`. For convenience, create the file with the contents:

```
DJANGO_DEBUG=true
```

Linting

We use a combination of flake8 and black for linting and formatting our code, respectively. To ensure that there are no formatting/linting errors on each commit, we use the pre-commit tool. To install it, simply run:

```
pipenv run pre-commit install
```

This will ensure that your code is formatted prior to every commit. If the tool is somehow circumvented or not run, our CI process also runs the same checks and will fail a build that has formatting or linting errors.

Dev Environment Overview

Quickstart

Clone the project and install the dependencies:

```
$ git clone git@github.com:knowmetools/km-api
$ cd km-api
$ pipenv install --dev
```

Local Dev Server

The development server can be run using the following command:

```
$ pipenv run km_api/manage.py runserver
```

Running Tests

Tests are run with pytest. To run the tests, make sure the requirements are installed and run the tests:

```
$ pipenv install --dev
$ pipenv run pytest km_api/
```

Building Docs

We use sphinx for building documentation, and the docs are automatically published using ReadTheDocs. If you want to build the docs locally, install the requirements and run the build command:

```
$ pipenv install -r requirements/docs.txt
$ cd docs
$ pipenv run make html
```

1.1.2 Release Cycle

Our release cycle and development process is based on A Successful Git Branching Model. To assist in the use of this model, we recommend the use of git-flow, which is a tool built on top of Git.

This document describes how we integrate the described branching model with GitHub, and the specifics of releasing new versions.

Note: We assume that `git-flow` has been installed and configured as described in *Project Setup*.

Normal Development

Most development should occur in the form of work on `feature/*` or `bugfix/*` branches. These branches are created from and merged back into the `develop` branch.

With regards to GitHub, no work should be done without the presence of an issue outlining the required work. This process makes it easy to ensure that branches (and the resulting pull requests) only address one thing at a time, resulting in easier code reviews.

Features

Features are the addition of new features or attributes that didn't exist before. To start a new feature:

```
$ git flow feature start my-feature-name
```

The created feature branch should be pushed to GitHub, and a pull request should be opened. This allows us to perform a code review and ensure that all automated tests pass. We encourage pull requests for feature branches to be opened early in the development process so that feedback can be provided sooner rather than later.

Once the feature is complete, tests have passed, and the code has been reviewed, someone with write permissions on the `develop` branch can run the following:

```
$ git flow feature finish my-feature-name
```

The completion of a feature is greatly assisted if the feature is first squashed down to one commit. This can be accomplished by the following:

```
$ git checkout develop
$ git pull
$ git checkout feature/my-feature-name
$ git rebase -i develop
```

When rebasing, simply squash down all the commits, and ensure that the final commit message describes the cummulative change.

Bug Fixes

Bug fixes should be used to correct unexpected or broken behavior that is not critical to the production environment. To start a new bugfix:

```
$ git flow bugfix start my-bugfix
```

The rest of the process is the same as the one for features.

New Releases

Releases are created based on the development branch. After creating a release branch, no new features should be added on to that release (but they can continue to be added to the `develop` branch). To start a new release:

```
$ git flow release start vX.Y.0
```

These releases should always have a 0 as the patch number. The creation of a release branch should be **IMMEDIATELY** followed by the bumping of the minor or major version using bumpversion:

```
$ bumpversion [minor|major]
```

Any additional tweaks or bug fixes required for the features that are a part of this release can continue to be added on to this release branch. The release can then be completed with:

```
$ git flow release finish vX.Y.0
```

During this command, you will be prompted for a message for the tag being created. This message should be `Release vX.Y.0`. After the command completes, make sure all impacted Git artifacts are pushed:

```
$ git push --tags && git push origin master && git push
```

Hotfixes

Hotfixes are used when there is a critical issue in production, and a fix cannot be delayed until the next release.

> **Warning:** Hotfixes should be used with caution because they typically do not go through the same pull request and review process that other changes do.

To start a new hotfix, first inspect the current version number. The hotfix branch should be created with the current version's patch number incremented by one. For example, if the current version is `v1.4.0`, the command would be:

```
$ git flow hotfix start v1.4.1
```

The creation of a hotfix branch should be **IMMEDIATELY** followed by the bumping of the patch number using bumpversion:

```
$ bumpversion patch
```

Once all relevant changes have been made and carefully tested, the hotfix can be completed and merged back:

```
$ git flow hotfix finish vX.Y.Z
```

The message for the newly created tag should be `Release vX.Y.Z`. After the process completes, make sure all Git artifacts are pushed:

```
$ git push --tags && git push origin master && git push
```

1.1.3 Deployment

For an example of how to deploy the application, see knowmetools/km-api-deployment.

Environment Variables

The following environment variables can be used to modify the application's behavior.

Note: The application will only attempt to use a Postgres database if all of `DJANGO_DB_HOST`, `DJANGO_DB_NAME`, `DJANGO_DB_PASSWORD`, `DJANGO_DB_PORT`, and `DJANGO_DB_USER` are set. If any of these settings are not provided, we fall back to a local SQLite database.

DJANGO_ALLOWED_HOSTS

Default: `''`

A comma separated list of allowed hosts.

Note: This must be set if `DJANGO_DEBUG` is set to `False`.

DJANGO_APPLE_RECEIPT_VALIDATION_ENDPOINT

Default: `https://sandbox.itunes.apple.com/verifyReceipt`

The endpoint used to verify subscription receipts from Apple. This can take one of two values:

- `https://sandbox.itunes.apple.com/verifyReceipt`
- `https://buy.itunes.apple.com/verifyReceipt`

DJANGO_AWS_REGION

Default: `us-east-1`

The AWS region to use for services such as S3 and SES.

DJANGO_DB_HOST

Default: `localhost`

The hostname of the Postgres database to connect to.

DJANGO_DB_NAME

Default: `''`

The name of the Postgres database to connect to.

DJANGO_DB_PASSWORD

Default:: `''`

The password of the user that the application connects to the Postgres database as.

DJANGO_DB_PORT

Default: `5432`

The port to connect to the Postgres database on.

DJANGO_DB_USER

Default: `''`

The name of the user to connect to the Postgres database as.

DJANGO_DEBUG

Default: `False`

Set to `True` (case insensitive) to enable Django's debug mode.

DJANGO_EMAIL_VERIFICATION_URL

Default: `https://example.com/verify/{key}'`

The template used to construct links for verifying a user's email address. The `{key}` portion of the template will be replaced with a unique token.

DJANGO_HTTPS

Default: `False`

Set to `True` (case insensitive) if the application is served over HTTPS.

DJANGO_IN_MEMORY_FILES

Default: `False`

Set to `True` (case insensitive) to store static files in memory. This is mainly used for testing.

DJANGO_MEDIA_ROOT

Default: `''`

The location on the server's filesystem to store user uploaded files at. This setting has no effect when `DJANGO_S3_STORAGE` is `True`.

DJANGO_PASSWORD_RESET_URL

Default: `https://example.com/reset/{key}`

The template used to construct password reset links. The `{key}` portion of the template will be replaced with a unique token.

DJANGO_S3_AWS_REGION

Default: `$DJANGO_AWS_REGION`

The AWS region that the S3 bucket used to store files is located in. Only takes effect when `DJANGO_S3_STORAGE` is `True`.

DJANGO_S3_BUCKET

Default: `''`

The name of the S3 bucket to store files in. Only takes effect when `DJANGO_S3_STORAGE` is `True`.

DJANGO_S3_STORAGE

Default: `False`

Set to `True` (case insensitive) to enable storage of static and user uploaded files in an S3 bucket. Requires the following settings to be provided:

- `DJANGO_S3_BUCKET`

DJANGO_SECRET_KEY

Default: `secret`

> **Warning:** The default value is only used if `DJANGO_DEBUG` is set to `True`. This is to avoid exposing a known secret key in a production environment.

The secret key that Django uses for a few security operations.

DJANGO_SENTRY_DSN

Default: `''`

The *Data Source Name* for the application's Sentry project. If provided logging of warnings and errors to Sentry is enabled.

DJANGO_SENTRY_ENVIRONMENT

Default: `default`

The name of the environment that should be provided as context when logging to Sentry. Only takes effect when `DJANGO_SENTRY_DSN` is provided.

DJANGO_SES_AWS_REGION

Default: `$DJANGO_AWS_REGION`

The AWS region to send SES emails from. Only takes effect when `DJANGO_SES_ENABLED` is `True`.

DJANGO_SES_ENABLED

Default: `False`

Set to `True` (case insensitive) to enable sending of emails using AWS SES.

DJANGO_STATIC_ROOT

Default: `''`

The location on the server's filesystem to store static files at. This setting has no effect when `DJANGO_S3_STORAGE` is `True`.

1.1.4 Changelog

v1.5.0

Features

- #379: Add endpoint to retrieve/update a user's Know Me subscription through Apple.

- #393: Allow users to verify their email address without their password.

Miscellaneous

- Completely changed the way that settings are provided. Rather than being specified in a separate python file that is then imported, behavior is now set through environment variables.
- #383: Remove MailChimp integration.
- #384: Remove CloudWatch integration.
- #386: Remove deployment configuration. Deployment is now handled by knowmetools/km-api-deployment.

v1.4.1

Bug Fixes

- #371: Restore journal owner's ability to delete user comments.

v1.4.0

Features

- #349: Customize email templates for account related actions.
- #364: Add admin view for listing users.

Bug Fixes

- #368: Fix incorrect app store link in invitation notification email.

v1.3.0

Features

- #236: Add email notifications when a user is invited to view another user's profiles.

v1.2.2

Fix issue where users could upgrade their own accounts to have staff permissions.

v1.2.1

Bug Fixes

- #357: Add user account image to Know Me user list response data.
- #358: Fix users not being able to delete accessors granting them access to another user's informtion.

v1.2.0

Features

- #354: Add Know Me user information to accessor response.

Bug Fixes

- #352: Remove duplicate entry from user list.

Miscellaneous

- #355: Bump package versions

v1.1.0

Features #346: Add endpoint to list previously accepted accessors. #347: Allow users granted access by an accessor to delete that accessor.

v1.0.2

Bug Fixes #341: Fix issue with list entries not being orderable. #343: Fix bug with duplicate rows when selecting Know Me users. #345: Fix issue with sharing errors not being caught and rendered.

v1.0.1

#337: Fix issue with API documentation crashing.

v1.0.0

Breaking Changes

- #296: Add separate endpoint to accept an accessor.
- #316: Paginate journal entries. The entries are now nested under the `results` key, and there is additional information returned such as the total number of entries and the URLs for the next and previous pages. Entries are listed in reverse chronological order.
- #332: Remove ability to manually create a Know Me user. As per #263, a Know Me user is automatically created for each registered user.

Features

- #233: Add config endpoint for Know Me app. It contains information such as the lowest useable iOS app version.
- #259: Add optional profile image for users.
- #263#277: Automatically create a Know Me user for each user. The Know Me user's image defaults to the user's profile image.
- #278: Include information about the user granted access through an accessor.
- #299: Add additional information to media resources. The resources can have a link instead of a file, and they have an integer to hint how they should be styled.
- #306: Add tracking of legacy users. The list of legacy users can be viewed/updated by staff.
- #313: Expose if a user is a staff member through the profile endpoint.
- #321: Allow media resources to be detached from profile items.
- #326: The Know Me user owned by the requesting user is guaranteed to be the first element in the list returned from `/know-me/users/`. Each user in the list also has a new `is_owned_by_current_user` boolean attribute.
- #328: Increase maximum upload size to 100MB.

v0.9.3

Bug Fixes

- #324: Fix handling of duplicate accessors.
- #325: Fix crash when listing Know Me user accessors as an unauthenticated user.
- #327: Return journal entry permissions at the list level, rather than only from the detail endpoint.
- #329: Add missing attribute indicating if a profile is private or not.

v0.9.2

#317: Fix error when attaching a media resource to a profile item.

v0.9.1

#276: Add missing journal entries URL to Know Me user information.

v0.9.0

Features

- #191#193#303: Allow profiles and their components to be manually ordered.
- #235: Add timestamps to remaining models.
- #300: Send the correct URLs in account emails.

Bug Fixes

- #271: Fix deployment failing with newly provisioned servers.
- #295: Fix MailChimp integration.

v0.8.0

Breaking Changes

- #253#282: Massive rewrite of profile features. This is a backwards incompatible change that modifies endpoints and the data returned from profile endpoints in addition to requiring a complete database wipe.
- #258: Require multiple calls to return full profile.
- #267: Refactor accessor permission fields. The permissions are now encompassed in a single `is_admin` field.

Features

- #246#247#251: Add endpoints for managing journal entries.
- #248#249#252: Add endpoints for managing comments on journal entries.
- #250: Add searching for journal entries.
- #267: All models related to the Know Me app have creation and last-updated timestamps.
- #276: Return URLs in Know Me user list rather than only from the detail endpoint.

Bug Fixes

- #254: Fix Ansible creating duplicate crontab entries.
- #261: Fix permissions on `KMUser` instances not respecting sharing.
- #262: Fix access to private profiles being too open.
- #265: Fix inability to tear down Terraform-provisioned infrastructure.

Miscellaneous

- #211: Remove old "emergency" models.
- #273: Document release process.

v0.7.1

Bug Fixes

- #244: Fix issue with registration serializer.

v0.7.0

Developed "Bookshelf" as a rework of the old "Gallery".

Breaking Changes

- Switched to third party account management package. This modified the endpoints used for email management, password resets, and registration. Also, the transition removed a lot of our account related models and modified their migrations. This means the existing database must be wiped and recreated.
- #209: Removed "emergency" related content.
- #214: Removed `is_default` field from profiles.
- #239: Removed Layer integration. This means Layer authentication tokens can no longer be obtained from the API.

Features

- #212#217: Added categories for media resources.
- #213: Add endpoint for listing media resources.

Bug Fixes

- #224: Fix issue with incorrect filtering of list views.

Miscellaneous

- #208: Switched to autogenerated documentation.

v0.6.1

Bug Fixes:

- #225: Fix production deployments being deployed to the staging environment.

v0.6.0

Implement sharing of profiles.

Features:

- #154: Allow profiles to be marked as private. Private profiles are not accessible by shared users unless they are explicitly granted access.

Bug Fixes:

- #142: Add missing migrations.
- #139#197: Add ability to share profiles with other users.
- #177: Fix issue with trying to register with an email address that is already being used.

v0.5.3

Update dependency versions.

v0.5.2

Bug Fixes

- #195: Fix issue with some endpoints only accepting a single-digit ID.

v0.5.1

Bug Fixes

- #143: Fix missing field on emergency contact admin page.

v0.5.0

This release was focused on renaming the components of a Know Me user's profile. As a result of this renaming, this release will break all existing data related to Know Me. This change also caused several endpoints to be renamed. The most relevant issue here is #65.

- `/know-me/gallery-items/*` to `/know-me/media-resources/*`
- `/know-me/profiles/*` to `/know-me/users/*`
- `/know-me/rows/*` to `/know-me/topics/*`

Breaking Changes

- #66: Remove grouped and paged row types.
- #85: Separated content for different types of profile items into different models. This means the data (other than name) from existing profile items is lost.

Features

- #67: Add emergency contacts for Know Me users.
- #68#100#112: Add list-type profile items.
- #79#91#96: Add ability for users to manage their emergency items.

Bug Fixes

- #70: Fixed regression in the error returned when attempting to log in with an unverified email address.
- #75: Fix parsing of JSON requests.

v0.4.0

Breaking Changes

- #27: Move user profile view from `/auth/profile/` to `/account/profile`.
- #36#54: Emails must be verified before being able to log in.
- #42: The user model was moved to the `account` app. This requires dropping any existing databases.

Features

- #28: Users can change their password.
- #34: Users can now request a password reset by email.
- #47: Allow users to manage their email addresses. They can now add/remove addresses and switch which one is the primary.
- #50: Users receive a notification when an email is added to their account.

Miscellaneous

- #41: Users can be authenticated by passing an `email` rather than a `username` to Django's `authenticate` function.
- #46: An admin user is created when the project is deployed.
- #52: Developers are no longer required to have a local settings file.

v0.3.0

Features

- #29#30#31: Automatically sync user info to a MailChimp list.

Miscellaneous

- #32: Ignore reports about disallowed hosts.

v0.2.0

Breaking Changes

- #18: Flattened URL structure.
- #21: Moved Layer authentication to the `/auth/layer/` endpoint.

Features

- #12: Add logging in production.
- #14: Refactor permissions implementation using `dry-rest-permissions` package.
- #19#20: Add documentation.

Bug Fixes

- #9: Ensure passwords are validated.

1.2 Indices and tables

- genindex
- modindex
- search