access_missouri Documentation

Release 3.2.2

Nathan Lawrence

May 11, 2021

Contents:

This is the technical documentation for Access Missouri's inner workings and mechanics. You can find the front-end for Access Missouri and more consumer-friendly documentation at https://www.accessmo.org.

CHAPTER 1

Access Missouri's Django Apps

As a Django project, Access Missouri is structured as a number of separate 'apps,' integrated into each other, but with discrete functionality.

1.1 general

If I were re-architecting Access Missouri tomorrow, we wouldn't have an app named "general." It's a minor disaster to manage.

Most of the logic and models stored in general are broad enough that they're dependencies for several other apps. General also houses person and organization models.

1.1.1 Models

Models in the general app should not be added to if it can be avoided. Ask Nathan for details.

AMBaseModel

This is the abstract model all Access Missouri models inherit from, which means all models get the following characteristics.

Note: `AMBaseModel` is intended to be an abstract model. Do not create instances of it, but make sure every single Access Missouri model does implement it. The canned functionality can save your bacon.

Model Fields:

- `created_at` - `DateTimeField` with the exact date/time a specific instance of a model was created at. Automatically fills. You shouldn't have to think about this.

- `updated_at` - `DateTimeField` with the exact date/time a specific instance of a model was last updated/changed at. Automatically fills. You shouldn't have to think about this.
- `extras` - Perhaps the most important of the inherited fields, this is a `JSONField` key-value store of anything that doesn't fit somewhere else in a model. Use liberally. More info in the database is better than less.

class general.models.**AMBaseModel**

Variables

- **created_at** (*DateTimeField*) – The exact date/time a specific instance of a model was created at. Automatically fills. You shouldn't have to think about this.
- **updated_at** (*DateTimeField*) – The exact date/time a specific instance of a model was last updated/changed at. Automatically fills. You shouldn't have to think about this.
- **extras** (*JSONField*) – Perhaps the most important of the inherited fields, this is a `JSONField` key-value store of anything that doesn't fit somewhere else in a model. Use liberally. More info in the database is better than less.

Person

The core model to all the people-centered functionality in Access Missouri. Surprisingly simple, actually.

class general.models.**Person**

Variables

- **first_name** (*CharField*) – Self explanatory. Max length 300.
- **last_name** (*CharField*) – Self explanatory. Max length 300.
- **middle_name** (*CharField*) – Self explanatory. Max length 300. Can be blank, cannot be `None`.
- **nickname** (*CharField*) – Pretty straightforward. If someone's name is Joan, but they go by "Peggy," this should be filled in, but also if someone's name is Robert and they go by "Bob." We use this for a lot of fuzzy object matching. Max length 300. Can be blank, cannot be `None`.
- **gender** (*CharField*) – Free-form `CharField`. Not really used. Can be blank, cannot be `None`.
- **suffix** (*CharField*) – "Jr.", "Sr.", etc. if relevant. Max length 10. Can be blank, cannot be `None`.
- **index_name** (*CharField*) – The least intuitive of the fields, this is a cached concatenation of the person's name fields designed to be easily queryable. Max length is 400.
- **flavor_text** (*CharField*) – Just like on a trading card, this is the text to give someone an idea of a person's character.

get_full_name()

Returns a person's formatted full name without making you do the heavy lifting in an inconsistent way.

Returns Full concatenated person name.

Return type str

get_absolute_url()

Concatenates absolute path (no "https://....", but "/path/to") to standard view of object.

Returns Path to person's standard template view.

Return type str

`get_admin_url()`

Concatenates absolute path (no "https://....", but "/path/to") to admin interface view of object.

Returns Path to person's standard Django admin UI view.

Return type str

1.1.2 Management Commands

As primary business logic, management tasks generally should not be stored in the general app.

---

person_full_name_to_index_name

A task that should eventually be deprecated and relegated to model managers and aggregation, this creates an easily-queryable version of someone's name and stores it in a searchable format.

Behind the scenes, what's happening is a bunch of conditional concatenation, then it's getting stored in the Person model's *index_name* field.

Example

Run the task:

```
$ python manage.py person_full_name_to_index_name
```

1.2 legislative

Models, logic, views and templates related to bill tracking and other legislative functions.

1.3 finance

Models, logic, views and templates related to finance information.

1.4 search

Mostly helper functions and classes dedicated to searching and API usage.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index