standalorm

Release 1.0.0

Jason Tolbert Jr

Jan 26, 2021

CONTENTS

standalorm is a Python library that enables you to harness the power of Django's ORM in standalone Python scripts.

CHAPTER

ONE

ORM INITIALIZATION

Provides a function for initializing standalorm.

standalorm.orm_init.**orm_init**(*file_dunder*)

Initializes standalorm. This function is the only thing from the library a typical end user should be importing into their code.

This function should ideally be called at the beginning of the user's code, but strictly speaking, only has to be called before the user imports their models. (To clarify, this function doesn't have to be called every single time a model is imported; just one call somewhere in the execution process is enough).

This function should never ever ever be called from models.py or it will probably break something.

Parameters **file_dunder** – orm_init() should always be called with the dunder variable __file__ as its sole argument. This is used to ascertain the directory from which the function is being called so standalorm knows where to look for the SQLite database, if one is in use. While this parameter isn't strictly necessary when a non-SQLite database is being used, passing it in regardless does no harm and it's easier for the end user to just always pass it in without having to worry about what database is being used.

CHAPTER

TWO

UTILITY FUNCTIONS

Various utility functions.

standalorm.utils.**get_connection_list**(*include_default: bool = True*, *current: bool = False*) → list

Gets a list of existing database connections.

Parameters

- **include_default** – If False, the default database connection will not be included in the returned list.
- **current** – If True, only the current database connection will be returned.

Returns A list of existing database connections.

standalorm.utils.**get_settings**() → dict

Gets standalorm's settings (orm-settings.toml, NOT to be confused with Django's settings.py).

Returns A dictionary of standalorm's settings.

standalorm.utils.**save_settings**()

Converts `orm_settings` into a TOML-formatted string which is then written to orm-settings.toml, overwriting its current contents.

standalorm.utils.**selection_prompt**(*prompt: str*, *choices: list*) → str

Prompts the user to select an option from an list of choices.

Parameters

- **prompt** – The text of the prompt.
- **choices** – A list of choices.

Returns The user's selection.

standalorm.utils.**set_pythonpath**(*path: str*)

Sets the PYTHONPATH environment variable to the value of `path`.

Parameters **path** – A filepath.

CHAPTER

THREE

DATABASE CONNECTION CREATORS

Interactive prompts for adding new database connections.

standalorm.db_makers.**env_config**() → dict
Configures a database connection using an environment variable (Oracle and PostgreSQL only).

Returns A dictionary containing information about the newly-created database connection.

standalorm.db_makers.**make_new_db**(*database*, *use_env*) → dict
Calls either oracle(), postgresql(), or sqlite() depending on the value of database.

Parameters

- **database** – The database for which a connection is going to be created ("oracle", "postgresql", or "sqlite").
- **use_env** – If True, standalorm will prompt the user to configure the connection using a URI environment variable (Oracle and PostgreSQL only).

Returns A dictionary containing information about the newly-created database connection.

standalorm.db_makers.**oracle**(*use_env: bool*) → dict
Creates a new Oracle database connection.

Parameters **use_env** – If True, the connection will be configured using a URI environment variable.

Returns A dictionary containing information about the newly-created database connection.

standalorm.db_makers.**postgresql**(*use_env: bool*) → dict
Creates a new PostgreSQL database connection.

Parameters **use_env** – If True, the connection will be configured using a URI environment variable.

Returns A dictionary containing information about the newly-created database connection.

standalorm.db_makers.**sqlite**() → dict
Create a new SQLite database connection.

Returns A dictionary containing information about the newly-created database connection.

CHAPTER

FOUR

COMMAND-LINE INTERFACE

4.1 standalorm

This is root of standalorm's command line interface.

```
standalorm [OPTIONS] COMMAND [ARGS]...
```

4.1.1 about

Display license and copyright information.

```
standalorm about [OPTIONS]
```

Options

-l, --license
Display the full license text.

4.1.2 db

Use standalorm's database connection management tools.

```
standalorm db [OPTIONS] COMMAND [ARGS]...
```

add

Add a new database connection.

DB is the kind of database you want to add a connection for (Oracle, PostgreSQL, or SQLite). You'll be prompted for this value if you don't specify it.

```
standalorm db add [OPTIONS] [DB]
```

Options

--env
Configure the connection using a URI environment variable (Oracle and PostgreSQL only.)

Arguments

DB
Optional argument

edit

Edit an existing database connection.

The settings for the connection you choose will be opened as a TXT file in your operating system's default editor for those kinds of files.

DB is the name of the database you want to edit. You'll be prompted for this value if you don't specify it.

```
standalorm db edit [OPTIONS] [DB]
```

Arguments

DB
Optional argument

ls

List existing database connections.

```
standalorm db ls [OPTIONS]
```

Options

-c, --current
List only the current connection.

remove

Remove an existing database connection.

If you remove your current connection, standalorm will automatically switch to the default SQLite connection.

DB is the name of the database connection you want to remove. You'll be prompted for this value if you don't specify it.

```
standalorm db remove [OPTIONS] [DB]
```

Arguments

DB
Optional argument

switch

Switch to a different database connection.

DB is the name of the connection you want to switch to. You'll be prompted for this value if you don't specify it.

```
standalorm db switch [OPTIONS] [DB]
```

Arguments

DB
Optional argument

4.1.3 makemigrations

Create database migrations.

```
standalorm makemigrations [OPTIONS]
```

4.1.4 migrate

Apply database migrations.

```
standalorm migrate [OPTIONS]
```

4.1.5 pycharm

Get information on enabling Django support in PyCharm.

```
standalorm pycharm [OPTIONS]
```

4.1.6 startapp

Create a Django app in your project's root directory.

APP_NAME is the name you want to assign to your app. If left empty, this will default to "db".

```
standalorm startapp [OPTIONS] [APP_NAME]
```

Options

-nc, --no-create
Don't create a new folder or the __init__.py and models.py files. Use this if you need to point standalorm to an existing app directory without creating a new one.

Arguments

APP_NAME
Optional argument

CHAPTER

FIVE

INSTALLATION

```
$ pip install standalorm
```

CHAPTER

SIX

SOURCE CODE

Those interested in standalorm's source code should see its GitHub repository.

CHAPTER

SEVEN

SUPPORT, BUG TRACKING, AND FEATURE REQUESTS

All support inquiries, bug reports, and feature requests should be directed to standalorm's GitHub issues page.

CHAPTER

EIGHT

ADDITIONAL NOTES

standalorm is intended for people who are already familiar with Django's ORM; as such, the basics of how to use the ORM are outside the scope of this documentation. If you're looking to familiarize yourself with Django's ORM, see Django's own documentation, particularly the sections on models and making queries.

CHAPTER

NINE

ATTRIBUTIONS

standalorm is based on Dan Caron's Django ORM Standalone Template.

"Django" is a registered trademark of the Django Software Foundation, which does not endorse this software.

CHAPTER

TEN

LICENSE

standalorm is released under the MIT License.

PYTHON MODULE INDEX

S

INDEX

Symbols

A

D

E

G

M

O

P

S