210HabitTracker

Jared Buchanan, Daniel Koohmarey, Fan Wu, Li Yang, Ted Schelb

Dec 14, 2020

CONTENTS

1 Indices and tables **5**

Python Module Index **7**

Index **9**

Habit Tracker Web Server entry point.

app.**add_activites**()
: Add activities for a habit.

app.**add_habit**()
: Add a habit.

app.**cur_user**()
: Test api for current_user.

app.**delete_habit**()
: Delete a habit.

    Parameters **str** (*habitname*) – name of habit to delete

app.**get_activites**()
: Get activities for a user.

app.**get_all_activites**()
: Get all activities for a user.

app.**get_habits**()
: Get habits for a user.

app.**load_user**(*username*)
: Retrive current user.

app.**login**()
: Login a user.

app.**logout**()
: Logout a user.

app.**register**()
: Register a user.

app.**render_habits**()
: Render habits page.

app.**render_login**()
: Render login page.

app.**render_progress**()
: Render progress page.

app.**test_login**(*username*, *password*)
: Route for test login.

app.**test_register**(*username*, *password*)
: Route for test register.

app.**unauthorized_callback**()
: Redirect to login page if not logged in.

Database manager for the habit server.

class db_manager.**DBManager**(*session*)
: Bases: object

    Database Manager for habit server.

    add_activity(*activity*)
    : Add a new activity log for a given user and habit.

Parameters **`UserActivity`** (*`activity`*) – activity to add

Return Result operation result, Ok or Err

`add_habit` (*habit*)
Add a habit to the databse for a particular user.

Parameters **`UserHabit`** (*`habit`*) – habit to add

Returns Result operation result, Ok or Err

`add_user` (*user*)
Add a new user to the database.

Parameters **`User`** (*`user`*) – user to add

Returns Result operation result, Ok or Err.

`delete_habit` (*habit*)
Delete habit for a particular user.

Parameters **`UserHabit`** (*`habit`*) – user habit to delete.

Returns Result operation result, Ok or Err

`does_habit_exist` (*habit*)
Check whether a given habit exists for a given user.

Parameters **`UserHabit`** (*`habit`*) – user habit to check existence for

Returns bool does the habit exist in the database?

`does_user_exist` (*username*)
Check that a username exist in the database.

Parameters **`str`** (*`username`*) – username to check.

Returns bool does user exist in database?

`get_activities` (*habit*, *trailing_days=100*)
Get all the activities for a particular habit.

Parameters

- **`UserHabit`** (*`habit`*) – grab all activities linked to this habit.
- **`Optional[int]`** (*`trailing_days`*) – filter the activities to last trailing_days number of days. If None, get all activities

Return Result operation result, Ok or Err

`get_activity_streak` (*habit*)
Get the current activity streak for a given habit.

Parameters **`UserHabit`** (*`habit`*) – user habit to get streak for

Return Result operation result, Ok or Err

`get_all_activities` (*user*, *trailing_days=100*)
Get all the activities for a particular habit.

Parameters **`User`** (*`user`*) – user to get habits from

Return Result operation result, Ok or Err

`get_habits` (*user*)
Get habits for a particular user.

Parameters **`User`** (`user`) – user to get habits from

Returns Result operation result, Ok or Err

Databse ORM models.

class `db_models.`**`User`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`, `flask_login.mixins.UserMixin`

Database ORM model representing a User.

`check_password`(*password*)

Check that hashed password matches expected hashed password.

Parameters **`str`** (`password`) – password to check

`get_id`()

Get unique id of user (just the username).

Return str user id

`hashed_password`

`set_password`(*password*)

Set hashed password for a user.

Parameters **`str`** (`password`) – password to hash and set.

`username`

class `db_models.`**`UserActivity`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`

Databse ORM model representing a single activity.

`habitname`

`id`

`timestamp`

`username`

class `db_models.`**`UserHabit`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`

Database ORM model representing a single user habit.

`habitname`

`username`

Habit Server Utilities.

class `utils.`**`AlchemyEncoder`**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: `json.encoder.JSONEncoder`

AlchemyEncoder for habit server.

`default`(*obj*)

Unwrap Result object and serialize to Json.

Parameters **`Result`** (`result`) – returned result of db_manager

Returns Json Json object

`utils.`**`get_activity_streak`**(*activities*, *current_date=datetime.datetime(2020, 12, 14, 13, 46, 24, 252265)*)

Get the current activity streak.

An activity streak is defined as the number of previous days in a row an activity has been logged.

Parameters **`List[UserActivity]`** (*`activities`*) – list of activities

Returns int current activity streak

`utils.`**`is_valid_email_addr`**(*addr*)

Validate whether an email address is syntactically valid.

Parameters **`str`** (*`addr`*) – email addres

Returns bool is valid?

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

app, **??**

d

db_manager, 1
db_models, 3

u

utils, 3

INDEX

A

add_activites() (*in module app*), 1
add_activity() (*db_manager.DBManager method*), 1
add_habit() (*db_manager.DBManager method*), 2
add_habit() (*in module app*), 1
add_user() (*db_manager.DBManager method*), 2
AlchemyEncoder (*class in utils*), 3
app
    module, 1

C

check_password() (*db_models.User method*), 3
cur_user() (*in module app*), 1

D

db_manager
    module, 1
db_models
    module, 3
DBManager (*class in db_manager*), 1
default() (*utils.AlchemyEncoder method*), 3
delete_habit() (*db_manager.DBManager method*), 2
delete_habit() (*in module app*), 1
does_habit_exist() (*db_manager.DBManager method*), 2
does_user_exist() (*db_manager.DBManager method*), 2

G

get_activites() (*in module app*), 1
get_activities() (*db_manager.DBManager method*), 2
get_activity_streak() (*db_manager.DBManager method*), 2
get_activity_streak() (*in module utils*), 3
get_all_activites() (*in module app*), 1
get_all_activities() (*db_manager.DBManager method*), 2
get_habits() (*db_manager.DBManager method*), 2
get_habits() (*in module app*), 1
get_id() (*db_models.User method*), 3

H

habitname (*db_models.UserActivity attribute*), 3
habitname (*db_models.UserHabit attribute*), 3
hashed_password (*db_models.User attribute*), 3

I

id (*db_models.UserActivity attribute*), 3
is_valid_email_addr() (*in module utils*), 4

L

load_user() (*in module app*), 1
login() (*in module app*), 1
logout() (*in module app*), 1

M

module
    app, 1
    db_manager, 1
    db_models, 3
    utils, 3

R

register() (*in module app*), 1
render_habits() (*in module app*), 1
render_login() (*in module app*), 1
render_progress() (*in module app*), 1

S

set_password() (*db_models.User method*), 3

T

test_login() (*in module app*), 1
test_register() (*in module app*), 1
timestamp (*db_models.UserActivity attribute*), 3

U

unauthorized_callback() (*in module app*), 1
User (*class in db_models*), 3
UserActivity (*class in db_models*), 3

UserHabit (*class in db_models*), 3
username (*db_models.User attribute*), 3
username (*db_models.UserActivity attribute*), 3
username (*db_models.UserHabit attribute*), 3
utils
    module, 3