
puffotter
Release 0.15.3

Hermann Krumrey

Sep 18, 2020

CONTENTS

1	puffotter	3
1.1	puffotter package	3
2	Indices and tables	23
	Python Module Index	25
	Index	27

Contents:

PUFFOTTER

1.1 puffotter package

1.1.1 Subpackages

puffotter.flask package

Subpackages

puffotter.flask.background package

Submodules

puffotter.flask.background.telegram module

`puffotter.flask.background.telegram.telegram_whoami()`

Specifies the background behaviour of the telegram bot By default, the bot listens to /whoami messages and answers with the telegram chat ID :return: None

Module contents

puffotter.flask.db package

Submodules

puffotter.flask.db.ApiKey module

class `puffotter.flask.db.ApiKey.ApiKey(*args, **kwargs)`

Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'api_keys' SQL table An ApiKey is used for API access using HTTP basic auth

`__init__(*args, **kwargs)`

Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

creation_time: int

The time at which this API key was created as a UNIX timestamp

has_expired () → bool
Checks if the API key has expired. API Keys expire after 30 days :return: True if the key has expired, False otherwise

id

key_hash: str
The hash of the API key

user: *puffotter.flask.db.User.User*
The user associated with this API key

user_id: int
The ID of the user associated with this API key

verify_key (key: str) → bool
Checks if a given key is valid :param key: The key to check :return: True if the key is valid, False otherwise

puffotter.flask.db.ModelMixin module

class puffotter.flask.db.ModelMixin.**ModelMixin**
Bases: object

A mixin class that specifies a couple of methods all database models should implement

id = Column(None, Integer(), table=None, primary_key=True, nullable=False)
The ID is the primary key of the table and increments automatically

puffotter.flask.db.TelegramChatId module

class puffotter.flask.db.TelegramChatId.**TelegramChatId** (*args, **kwargs)
Bases: *puffotter.flask.db.ModelMixin.ModelMixin*, sqlalchemy.ext.declarative.api.Model

Model that describes the 'telegram_chat_ids' SQL table Maps telegram chat ids to users

__init__ (*args, **kwargs)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

chat_id: str
The telegram chat ID

id

send_message (message_text: str)
Sends a message to the telegram chat :param message_text: The message text to send :return: None

user: *puffotter.flask.db.User.User*
The user associated with this telegram chat ID

user_id: int
The ID of the user associated with this telegram chat ID

puffotter.flask.db.User module

class puffotter.flask.db.User.**User** (*args, **kwargs)

Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'users' SQL table A User stores a user's information, including their email address, username and password hash

__init__ (*args, **kwargs)

Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

api_keys: **List**[**ApiKey**]

API keys for this user

confirmation_hash: **str**

The account's confirmation hash. This is the hash of a key emailed to the user. Only once the user follows the link in the email containing the key will their account be activated

confirmed: **bool**

The account's confirmation status. Logins should be impossible as long as this value is False.

email: **str**

The user's email address

get_id() → **str**

Method required by flask-login :return: The user's ID as a unicode string

id

property is_active

Property required by flask-login :return: True

property is_anonymous

Property required by flask-login :return: True if the user is not confirmed, False otherwise

property is_authenticated

Property required by flask-login :return: True if the user is confirmed, False otherwise

password_hash: **str**

The user's hashed password, salted and hashed.

telegram_chat_id: **Optional**[**TelegramChatId**]

Telegram chat ID for the user if set up

username: **str**

The user's username

verify_confirmation (*confirmation_key*: *str*) → **bool**

Verifies a confirmation key against the confirmation hash :param confirmation_key: The key to check :return: True if the key matches, False otherwise

verify_password (*password*: *str*) → **bool**

Verifies a password against the password hash :param password: The password to check :return: True if the password matches, False otherwise

Module contents

puffotter.flask.routes package

Subpackages

puffotter.flask.routes.api package

Submodules

puffotter.flask.routes.api.user_management module

```
puffotter.flask.routes.api.user_management.define_blueprint (blueprint_name:  
                                                                str) →  
                                                                flask.blueprints.Blueprint
```

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

Module contents

Submodules

puffotter.flask.routes.decorators module

```
puffotter.flask.routes.decorators.api (func: Callable) → Callable
```

Decorator that handles common API patterns and ensures that the JSON response will always follow a certain pattern :param func: The function to wrap :return: The wrapper function

```
puffotter.flask.routes.decorators.api_login_required (func: Callable) → Callable
```

Decorator to make unauthorized API calls respond with JSON properly :param func: The function to wrap :return: The wrapped function

puffotter.flask.routes.static module

```
puffotter.flask.routes.static.define_blueprint (blueprint_name: str) →  
                                                flask.blueprints.Blueprint
```

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

puffotter.flask.routes.user_management module

```
puffotter.flask.routes.user_management.define_blueprint (blueprint_name: str) →  
                                                            flask.blueprints.Blueprint
```

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

Module contents

`puffotter.flask.routes.blueprint_generators`: `List[Tuple[Callable[str, flask.blueprints.Blueprint], str]]`
 Defines the functions used to create the various blueprints as well as their names

puffotter.flask.test package

Subpackages

puffotter.flask.test.db package

Submodules

puffotter.flask.test.db.TestApiKey module

class `puffotter.flask.test.db.TestApiKey.TestApiKey` (*methodName='runTest'*)

Bases: `puffotter.flask.test.TestFramework._TestFramework`

Class that tests the ApiKey database model

test_equality ()

Tests checking equality for model objects :return: None

test_expiration ()

Tests if the expiration of API keys works correctly :return: None

test_hashing ()

Tests using the model objects as keys in a dictionary :return: None

test_json_representation ()

Tests the JSON representation of the model :return: None

test_repr ()

Tests the `__repr__` method of the model class :return: None

test_string_representation ()

Tests the string representation of the model :return: None

test_verifying_key ()

Tests verifying an api key :return: None

puffotter.flask.test.db.TestModelMixin module

class `puffotter.flask.test.db.TestModelMixin.TestModelMixin` (*methodName='runTest'*)

Bases: `puffotter.flask.test.TestFramework._TestFramework`

Class that tests the ModelMixin class

test_enum_attributes ()

Tests if the repr method handles enums correctly :return: None

puffotter.flask.test.db.TestUser module

```
class puffotter.flask.test.db.TestUser.TestUser (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework

    Class that tests the User database model

    test_equality()
        Tests checking equality for model objects :return: None

    test_flask_properties()
        Tests if the flask_login properties work as expected :return: None

    test_hashing()
        Tests using the model objects as keys in a dictionary :return: None

    test_json_representation()
        Tests the JSON representation of the model :return: None

    test_json_with_telegram_chat_id()
        Tests whether or not including children can cause recursion errors :return: None

    test_repr()
        Tests the __repr__ method of the model class :return: None

    test_string_representation()
        Tests the string representation of the model :return: None

    test_verifying_password()
        Tests verifying the password of a user :return: None
```

Module contents

puffotter.flask.test.misc package

Submodules

puffotter.flask.test.misc.TestApiCalls module

```
class puffotter.flask.test.misc.TestApiCalls.TestApiCalls (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework

    Class that tests various API calls

    test_exception_in_api_route()
        Tests if an exception on an API route is correctly wrapped in a JSON response :return: None

    test_expired_api_key()
        Tests using an expired API key :return: None

    test_non_base64_header()
        Tests using a header that's not base64 encoded :return: None

    test_random_exception()
        Tests that the API routes catch any Exceptions without issue :return: None

    test_unauthorized_call()
        Tests and unauthorized API call :return: None
```

test_using_non_json_data()
 Tests sending the data as something that's not JSON :return: None

puffotter.flask.test.misc.TestConfig module

class puffotter.flask.test.misc.TestConfig.**TestConfig** (*methodName='runTest'*)
 Bases: puffotter.flask.test.TestFramework._TestFramework
 Class that tests the config class

test_db_config()
 Tests the database configuration :return: None

test_version()
 Tests if the version is fetched correctly :return: None

puffotter.flask.test.misc.TestErrorHandling module

class puffotter.flask.test.misc.TestErrorHandling.**TestErrorHandling** (*methodName='runTest'*)
 Bases: puffotter.flask.test.TestFramework._TestFramework
 Class that tests the flask error handling

test_404()
 Tests if a 404 error is handled correctly :return: None

test_exception()
 Tests if unexpected exceptions are caught correctly :return: None

puffotter.flask.test.misc.TestInitialization module

class puffotter.flask.test.misc.TestInitialization.**TestInitialization** (*methodName='runTest'*)
 Bases: puffotter.flask.test.TestFramework._TestFramework
 Tests the initialization of the flask application

test_missing_environment_variables()
 Tests if missing environment variables are detected correctly :return: None

test_missing_required_template()
 Tests if missing template files are detected correctly :return: None

puffotter.flask.test.misc.TestWsgi module

class puffotter.flask.test.misc.TestWsgi.**TestWsgi** (*methodName='runTest'*)
 Bases: puffotter.flask.test.TestFramework._TestFramework
 Tests the WSGI server

test_starting_and_stopping_wsgi_server()
 Tests starting and stopping the WSGI server :return: None

test_starting_background_tasks()
 Tests starting background tasks :return: None

Module contents

puffotter.flask.test.routes package

Subpackages

puffotter.flask.test.routes.api package

Submodules

puffotter.flask.test.routes.api.TestApiKeyRoute module

```
class puffotter.flask.test.routes.api.TestApiKeyRoute.TestApiKeyRoute (methodName='runTest')  
    Bases: puffotter.flask.test.TestFramework._TestFramework
```

Class that tests API-key related features

```
test_requesting_api_key ()  
    Tests requesting an API key :return: None
```

```
test_requesting_invalid_api_keys ()  
    Tests requesting API keys with invalid data :return: None
```

```
test_revoking_api_key ()  
    Tests revoking an API key :return: None
```

```
test_unsuccessfully_revoking_api_key ()  
    Tests unsuccessfully revoking an API key :return: None
```

Module contents

Submodules

puffotter.flask.test.routes.TestForgotRoute module

```
class puffotter.flask.test.routes.TestForgotRoute.TestForgotRoute (methodName='runTest')  
    Bases: puffotter.flask.test.TestFramework._TestFramework
```

Class that tests password reset features

```
test_invalid_recaptcha ()  
    Tests that invalid ReCaptcha responses are handled correctly :return: None
```

```
test_page_get ()  
    Tests getting the page :return: None
```

```
test_resetting_password ()  
    Tests successfully resetting a password :return: None
```

```
test_unsuccessfully_resetting_password ()  
    Tests unsuccessfully resetting a password :return: None
```

puffotter.flask.test.routes.TestLoginRoute module

```
class puffotter.flask.test.routes.TestLoginRoute.TestLoginRoute (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework

    Class that tests log-in features

    test_invalid_login_attempts ()
        Tests trying to log in with invalid credentials etc :return: None

    test_logging_in_and_out ()
        Tests logging in successfully, then once more, then logging out :return: None

    test_page_get ()
        Tests getting the page :return: None
```

puffotter.flask.test.routes.TestProfileRoute module

```
class puffotter.flask.test.routes.TestProfileRoute.TestProfileRoute (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework

    Class that tests profile features

    test_changing_password ()
        Tests changing a password :return: None

    test_page_get ()
        Tests getting the page :return: None

    test_unsuccessful_password_change ()
        Tests unsuccessfully changing a password :return: None

    test_unsuccessful_user_delete ()
        Tests unsuccessfully deleting a user :return: None

    test_user_delete ()
        Tests deleting a user :return: None
```

puffotter.flask.test.routes.TestRegisterRoute module

```
class puffotter.flask.test.routes.TestRegisterRoute.TestRegisterRoute (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework

    Class that tests registration features

    test_confirming ()
        Tests confirming a user :return: None

    test_invalid_confirm ()
        Tests invalid confirmations :return: None

    test_invalid_recaptcha ()
        Tests that invalid ReCaptcha responses are handled correctly :return: None

    test_invalid_registrations ()
        Tests registering using invalid parameters :return: None

    test_page_get ()
        Tests getting the page :return: None
```

```
test_registering_user ()
    Tests registering a new user :return: None
```

puffotter.flask.test.routes.TestStaticRoutes module

```
class puffotter.flask.test.routes.TestStaticRoutes.TestStaticRoutes (methodName='runTest')
    Bases: puffotter.flask.test.TestFramework._TestFramework
```

Class that tests static pages

```
test_get_about ()
    Tests getting the about page :return: None
```

```
test_get_index ()
    Tests getting the index page :return: None
```

```
test_get_privacy ()
    Tests getting the privacy page :return: None
```

Module contents

Submodules

puffotter.flask.test.TestFramework module

Module contents

Submodules

puffotter.flask.Config module

```
class puffotter.flask.Config.Config
    Bases: object
```

Class that keeps track of configuration data The class attributes should only be called after running load_config

```
API_VERSION: str = '1'
    The API Version
```

```
BEHIND_PROXY: bool
    Whether or not the site is being served by a reverse proxy like nginx.
```

```
DB_MODE: str
    The database mode (for example 'sqlite' or 'mysql')
```

```
DB_URI: str
    The database URI to use for database operations
```

```
DEBUG_LOGGING_PATH: str
    The path to the debug logging path
```

```
DOMAIN_NAME: str
    The domain name of the website
```

```
FLASK_SECRET: str
    The flask secret key
```

HTTP_PORT: int
The port to use when serving the flask application

LOGGING_PATH: str
The path to the logging file

MAX_API_KEY_AGE: int = 2592000
The maximum age for API keys

MAX_USERNAME_LENGTH: int = 12
The maximum length of usernames

MIN_USERNAME_LENGTH: int = 1
The Minimum length for usernames

RECAPTCHA_SECRET_KEY: str
Google ReCaptcha secret key for bot detection

RECAPTCHA_SITE_KEY: str
Google ReCaptcha site key for bot detection

REQUIRED_TEMPLATES: Dict[str, str] = {'about': 'static/about.html', 'error_page': 's
Specifies required template files

SENTRY_DSN: str
The sentry DSN used for error logging

SMTP_ADDRESS: str
The SMPT Address used for sending emails

SMTP_HOST: str
The SMPT Host used for sending emails

SMTP_PASSWORD: str
The SMPT Password used for sending emails

SMTP_PORT: int
The SMPT Port used for sending emails

STRINGS: Dict[str, str] = {'401_message': 'You are not logged in', '500_message': 'T
Dictionary that defines various strings used in the application. Makes it easier to use custom phrases.

TELEGRAM_API_KEY: str
API key for a telegram bot

TELEGRAM_BOT_CONNECTION: bokkichat.connection.impl.TelegramBotConnection.TelegramBotCo
Telegram bot connection

TELEGRAM_WHOAMI: bool
Whether or not the telegram WHOAMI background thread will be started

TEMPLATE_EXTRAS: Dict[str, Callable[Dict[str, Any]]] = {'about': <function Config.<lar
This can be used to provide the template rendering engine additional parameters, which may be necessary when adding UI elements. This is done with functions that don't expect any input and return a dictionary of keys and values to be passed to the template rendering engine

TESTING: bool
Whether or not testing is enabled

VERBOSITY: int
The verbosity level of the logging when printing to the console

VERSION: str
The current version of the application

WARNING_LOGGING_PATH: `str`

The path to the logging path for WARNING messages

classmethod `base_url` () → `str`

Returns The base URL of the website

classmethod `dump_env_variables` (*path: Optional[str] = None*)

Dumps all environment variables used by this application to a file `:param path`: The path to the file to which to dump the content.

If this is `None`, the file contents will be printed.

Returns `None`

classmethod `ensure_environment_variables_present` ()

Makes sure that all required environment variables have been set. If this is not the case, the app will exit.
`:return`: `None`

classmethod `environment_variables` () → `Dict[str, List[str]]`

Specifies required and optional environment variables `:return`: The specified environment variables in two lists in

a dictionary, grouped by whether the variables are required or optional

classmethod `initialize_telegram` ()

Initializes the telegram bot connection `:return`: `None`

classmethod `load_config` (*root_path: str, module_name: str, sentry_dsn: str*)

Loads the configuration from environment variables `:param root_path`: The root path of the application
`:param module_name`: The name of the project's module `:param sentry_dsn`: The sentry DSN used for error logging `:return`: `None`

puffotter.flask.base module

`puffotter.flask.base.app` = `<Flask 'puffotter.flask.base'>`

The Flask App

`puffotter.flask.base.db` = `<SQLAlchemy engine=None>`

The SQLAlchemy database connection

`puffotter.flask.base.login_manager` = `<flask_login.login_manager.LoginManager object>`

The Flask-Login Login Manager

puffotter.flask.enums module

class `puffotter.flask.enums.AlertSeverity` (*value*)

Bases: `enum.Enum`

Enumeration that defines the various levels of severity an alert can have

DANGER = `'danger'`

Translates to a red alert

INFO = `'info'`

Translates to a blue alert

SUCCESS = `'success'`

Translates to a green alert

```
WARNING = 'warning'
    Translates to a yellow alert
```

puffotter.flask.exceptions module

```
exception puffotter.flask.exceptions.ApiException(reason: str, status_code: int)
    Bases: Exception
```

Api raised when an API-related exception occurs

```
__init__(reason: str, status_code: int)
    Initializes the exception :param reason: The reason the API Exception was raised :param status_code: The
    status code associated with the exception
```

puffotter.flask.initialize module

```
puffotter.flask.initialize.CREATED_BLUEPRINTS = []
    Keeps track of created blueprint names. This is necessary for unit testing with nose, because duplicate blueprint
    names will cause errors.
```

```
puffotter.flask.initialize.init_flask(module_name: str, sentry_dsn: str, root_path: str,
    config: Type[puffotter.flask.Config.Config], models:
    List[Type[sqlalchemy.ext.declarative.api.Model]],
    blueprint_generators: List[Tuple[Callable[str,
    flask.blueprints.Blueprint], str]], extra_jinja_vars:
    Optional[Dict[str, Any]] = None)
```

Initializes the flask application :param module_name: The name of the module :param sentry_dsn: The sentry DSN used for error logging :param root_path: The root path of the flask application :param config: The Config class to use for configuration :param models: The database models to create :param blueprint_generators: Tuples that contain a function that generates

a blueprint and the name of the blueprint

Parameters `extra_jinja_vars` – Any extra variables to pass to jinja

Returns None

puffotter.flask.wsgi module

```
puffotter.flask.wsgi.start_server(config: Type[puffotter.flask.Config.Config],
    task_definitions: Dict[str, Tuple[int, Callable]])
```

Starts the flask application using a cheroot WSGI server :param config: The configuration to use :param task_definitions: The background tasks, consisting of:

- the name of the task
- the time to wait before running the task again
- a function that takes no arguments that executes the task

Returns None

Module contents

puffotter.test package

Submodules

puffotter.test.TestCrypto module

```
class puffotter.test.TestCrypto.TestCrypto (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Tests cryptographical functions  
test_hashing()  
    Tests that passwords can be hashed successfully :return: None  
test_hashing_strings()  
    Tests that password hashing and verifying works with string as well :return: None  
test_salt()  
    Tests that password hashes are salted :return: None  
test_verifying_with_invalid_hash()  
    Tests that attempting to verify a password with an incorrectly formatted hash will return False :return:  
    None
```

puffotter.test.TestOs module

```
class puffotter.test.TestOs.TestCrypto (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Tests os functions  
test_listdir()  
    Tests the listdir function :return: None
```

puffotter.test.TestUnits module

```
class puffotter.test.TestUnits.TestCrypto (methodName='runTest')  
    Bases: unittest.case.TestCase  
    Tests cryptographical functions  
test_converting_bytes_to_human_readable()  
    Tests that passwords can be hashed successfully :return: None
```

Module contents

1.1.2 Submodules

1.1.3 puffotter.crypto module

`puffotter.crypto.generate_hash` (*password: str*) → str

Salts and hashes a password to generate a hash for storage in a database :param password: The password to hash
:return: The hash of the password

`puffotter.crypto.generate_random` (*length: int*) → str

Generates a random byte string consisting of alphanumeric characters Thanks @ Albert (<https://stackoverflow.com/users/281021/albert>) <https://stackoverflow.com/questions/2257441> :param length: The length of the string to generate :return: The generated random byte string

`puffotter.crypto.verify_password` (*password: str, hashed: str*)

Verifies that a password matches a given hash :param password: The password to verify :param hashed: The hash to verify the password against :return: True if the password matches, otherwise False

1.1.4 puffotter.env module

`puffotter.env.load_env_file` (*path: str = '.env'*)

Loads an environment file into `os.environ` :param path: The path to the environment file :return: None

1.1.5 puffotter.graphql module

class `puffotter.graphql.GraphQLClient` (*api_url: str*)

Bases: object

A simple API wrapper for GraphQL APIs

__init__ (*api_url: str*)

Initializes the GraphQL API wrapper :param api_url: The API endpoint URL

query (*query_string: str, variables: Optional[Dict[str, Any]]*) → Optional[Dict[str, Any]]

Executes a GraphQL query :param query_string: The query string to use :param variables: The variables to send :return: The response JSON, or None if an error occurred.

1.1.6 puffotter.imap module

`puffotter.imap.get_inbox_count` (*imap_server: str, imap_address: str, imap_password: str, imap_port: int = 993*) → int

Checks the amount of emails in an IMAP inbox :param imap_server: The IMAP server to use :param imap_address: The IMAP address to use :param imap_password: The IMAP password to use :param imap_port: The IMAP port to use :return: The amount of emails

1.1.7 puffotter.init module

`puffotter.init.argparse_add_logfile` (*parser: argparse.ArgumentParser*)

Adds the `-logfile` argument to the argument parser :param parser: The argument parser to modify :return: None

`puffotter.init.argparse_add_verbosity` (*parser: argparse.ArgumentParser*)

Adds `-quiet`, `-verbose` and `-debug` parameters to an ArgumentParser :param parser: the parser to which to add those flags :return: None

`puffotter.init.cli_start` (*main_func: Union[Callable[None], Callable[argparse.Namespace, None], Callable[[argparse.Namespace, logging.Logger], None]], arg_parser: argparse.ArgumentParser, exit_msg: str = 'Goodbye', package_name: Optional[str] = None, sentry_dsn: Optional[str] = None, release_name: Optional[str] = None*)

Starts a program and sets up logging, as well as sentry error tracking :param main_func: The main function to call :param arg_parser: The argument parser to use :param exit_msg: The message printed when the program's execution is

stopped using a keyboard interrupt

Parameters

- **package_name** – The package name of the application
- **sentry_dsn** – The sentry DSN to use
- **release_name** – The name of the release

Returns None

`puffotter.init.setup_logging` (*args: argparse.Namespace, package_name: Optional[str]*)

Sets up logging for the provided arguments :param args: The CLI arguments :param package_name: The package name :return: None

1.1.8 puffotter.json module

`puffotter.json.jsonify_models` (*data: Dict[str, Any], deep: bool = True*) → Dict[str, Any]

Serializes a dictionary and calls the `__json__()` method on all objects that have one. :param data: The data to serialize :param deep: Indicates whether or not child models are included.

If False, only IDs will be included.

Returns The serialized data

1.1.9 puffotter.logging module

class `puffotter.logging.ColorLogger` (*logger: logging.Logger, debug_bg: str = '\x1b[47m', debug_fg: str = '\x1b[30m', info_bg: str = '\x1b[40m', info_fg: str = '\x1b[37m', warning_bg: str = '\x1b[43m', warning_fg: str = '\x1b[30m', error_bg: str = '\x1b[101m', error_fg: str = '\x1b[30m'*)

Bases: object

__init__ (*logger: logging.Logger, debug_bg: str = '\x1b[47m', debug_fg: str = '\x1b[30m', info_bg: str = '\x1b[40m', info_fg: str = '\x1b[37m', warning_bg: str = '\x1b[43m', warning_fg: str = '\x1b[30m', error_bg: str = '\x1b[101m', error_fg: str = '\x1b[30m'*)

Initialize self. See help(type(self)) for accurate signature.

debug (*message: str, bg: Optional[str] = None, fg: Optional[str] = None*)

Logs a message at DEBUG level :param message: The message to log :param bg: Overrides the default background style :param fg: Overrides the default foreground style :return: None

error (*message: str, bg: Optional[str] = None, fg: Optional[str] = None*)

Logs a message at ERROR level :param message: The message to log :param bg: Overrides the default background style :param fg: Overrides the default foreground style :return: None

info (*message: str, bg: Optional[str] = None, fg: Optional[str] = None*)

Logs a message at INFO level :param message: The message to log :param bg: Overrides the default background style :param fg: Overrides the default foreground style :return: None

warning (*message: str, bg: Optional[str] = None, fg: Optional[str] = None*)

Logs a message at WARNING level :param message: The message to log :param bg: Overrides the default background style :param fg: Overrides the default foreground style :return: None

1.1.10 puffotter.os module

`puffotter.os.create_file` (*path: str*)

Creates an empty file :param path: The path to the file :return: None

`puffotter.os.get_ext` (*filename: str*) → Optional[str]

Gets the file extension of a file :param filename: The filename for which to get the file extension :return: The file extension or None if the file has no extension

`puffotter.os.listdir` (*path: str, no_files: bool = False, no_dirs: bool = False, no_dot: bool = True*)
→ List[Tuple[str, str]]

Improves on the standard `os.listdir` function. By default, files and directories starting with `.` are ignored and one can disable checking for files or directories. Instead of just the filenames, the function returns tuples of filenames and relative file paths. :param path: The path of which to list the contents :param no_files: If set to True, will ignore files :param no_dirs: If set to True, will ignore directories :param no_dot: If set to True, will ignore files starting with `.` :return: A sorted list of the contents of the directory, consisting of

tuples of the file/directory name and their relative path.

`puffotter.os.makedirs` (*path: str, delete_before: bool = False*)

A more pleasant to use `makedirs` function. Only calls `os.makedirs` if the directory does not already exist :param path: The path to the directory to create :param delete_before: If True, deletes the directory beforehand, if it exists

Returns None

`puffotter.os.replace_illegal_ntfs_chars` (*string: str*) → str

Replaces illegal characters in NTFS file systems with appropriate substitutes :param string: The string in which to replace the illegal characters :return: The sanitized string

1.1.11 puffotter.print module

`puffotter.print.pprint` (*objects: str, sep: str = ' ', end: str = '\n', file: IO = <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>, flush: bool = False, fg: Optional[str] = None, bg: Optional[str] = None)
Function that extends the print function :return: None

1.1.12 puffotter.prompt module

`puffotter.prompt.prompt` (prompt_text: str = "", default: Optional[Any] = None, _type: Callable[str, Any] = <class 'str'>, required: bool = True, choices: Optional[Set[str]] = None) → Optional[Any]

Generic prompt with configuration options :param prompt_text: The text to display before the prompt :param default: A default value to use if the user responds with '' :param _type: The type of the object prompted. Must take a single string

as a parameter

Parameters

- **required** – Whether or not a response is required
- **choices** – Valid choices for the prompt

Returns The prompt result. May be None if required is False

`puffotter.prompt.prompt_comma_list` (message: str, primitive_type: Callable[str, Any] = <class 'str'>, min_count: int = 0, no_empty: bool = True, default: Optional[List[Any]] = None) → List[Any]

Prompts the user for a comma-separated list :param message: The message to display :param primitive_type: The primitive type of the elements in the list :param min_count: The minimum amount of elements to be provided by the user :param no_empty: Removes any empty strings :param default: A default value :return: The result of the prompt

`puffotter.prompt.selection_prompt` (objects: List[object]) → List[object]

Prompts the user for a selection from a list of objects :param objects: The objects to show :return: The selection of objects

`puffotter.prompt.yn_prompt` (message: str, make_sure: bool = True, case_sensitive: bool = False) → bool

Creates a yes/no prompt :param message: The message to display :param make_sure: Continuously prompts if the response is neither

'y' or 'n' until it is. If false, every input besides 'y' will result in the return value being False

Parameters **case_sensitive** – Whether or not the prompt should be case-sensitive

Returns True if the user specified 'y', else False

1.1.13 puffotter.recaptcha module

`puffotter.recaptcha.verify_recaptcha` (*client_ip: str, recaptcha_response: str, secret_key: str*)
→ bool

Verifies a recaptcha response. If the recaptcha response originates from a local address, this method will always return True. :param client_ip: The IP Address of the client solving the captcha :param recaptcha_response: the recaptcha response to verify :param secret_key: the recaptcha secret key :return: True if the recaptcha response was correct, False otherwise

1.1.14 puffotter.requests module

`puffotter.requests.aggressive_request` (*url: str*) → str

Handles GET requests while analyzing status codes :param url: The URL to get :return: The response text

1.1.15 puffotter.smtp module

`puffotter.smtp.send_email` (*address: str, title: str, message: str, smtp_server: str, smtp_address: str,*
smtp_password: str, smtp_port: int = 587)

Sends an HTML email message using SMTP :param address: The address to send to :param title: The email's title :param message: The message to send :param smtp_server: The SMTP server to use :param smtp_address: The SMTP address to use :param smtp_password: The SMTP password to use :param smtp_port: The SMTP port to use :return: None

1.1.16 puffotter.subprocess module

`puffotter.subprocess.execute_command` (*command: List[str]*) → int

Executes a command :param command: The command to execute :return: The status code

1.1.17 puffotter.testutils module

1.1.18 puffotter.units module

`puffotter.units.byte_string_to_byte_count` (*byte_string: str*) → int

Converts a string representing bytes to a number of bytes. For example: "500K" -> 500 000

"2.5M" -> 2 500 000 "10GB" -> 10 000 000 000 "30kb/s" -> 30 000

Parameters `byte_string` – The string to convert

Returns The amount of bytes

`puffotter.units.human_readable_bytes` (*bytecount: int, base_1024: bool = False, re-*
move_trailing_zeroes: bool = True) → str

Converts an amount of bytes into a human-readable string :param bytecount: The bytes to convert :param base_1024: Whether or not to use 1024 as base (for mebibytes etc) :param remove_trailing_zeroes: If set to True, will remove any trailing

zeroes from the string

Returns The human-readable string

1.1.19 Module contents

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

- puffotter, 22
- puffotter.crypto, 17
- puffotter.env, 17
- puffotter.flask, 16
 - puffotter.flask.background, 3
 - puffotter.flask.background.telegram, 3
 - puffotter.flask.base, 14
 - puffotter.flask.Config, 12
 - puffotter.flask.db, 6
 - puffotter.flask.db.ApiKey, 3
 - puffotter.flask.db.ModelMixin, 4
 - puffotter.flask.db.TelegramChatId, 4
 - puffotter.flask.db.User, 5
 - puffotter.flask.enums, 14
 - puffotter.flask.exceptions, 15
 - puffotter.flask.initialize, 15
 - puffotter.flask.routes, 7
 - puffotter.flask.routes.api, 6
 - puffotter.flask.routes.api.user_management, 6
 - puffotter.flask.routes.decorators, 6
 - puffotter.flask.routes.static, 6
 - puffotter.flask.routes.user_management, 6
 - puffotter.flask.test, 12
 - puffotter.flask.test.db, 8
 - puffotter.flask.test.db.TestApiKey, 7
 - puffotter.flask.test.db.TestModelMixin, 7
 - puffotter.flask.test.db.TestUser, 8
 - puffotter.flask.test.misc, 10
 - puffotter.flask.test.misc.TestApiCalls, 8
 - puffotter.flask.test.misc.TestConfig, 9
 - puffotter.flask.test.misc.TestErrorHandling, 9
 - puffotter.flask.test.misc.TestInitialization, 9
 - puffotter.flask.test.misc.TestWsgi, 9
 - puffotter.flask.test.routes, 12
 - puffotter.flask.test.routes.api.TestApiKeyRoute, 10
 - puffotter.flask.test.routes.TestForgotRoute, 10
 - puffotter.flask.test.routes.TestLoginRoute, 11
 - puffotter.flask.test.routes.TestProfileRoute, 11
 - puffotter.flask.test.routes.TestRegisterRoute, 11
 - puffotter.flask.test.routes.TestStaticRoutes, 12
 - puffotter.flask.test.TestFramework, 12
 - puffotter.flask.wsgi, 15
 - puffotter.graphql, 17
 - puffotter.imap, 17
 - puffotter.init, 18
 - puffotter.json, 18
 - puffotter.logging, 18
 - puffotter.os, 19
 - puffotter.print, 20
 - puffotter.prompt, 20
 - puffotter.recaptcha, 21
 - puffotter.requests, 21
 - puffotter.smtp, 21
 - puffotter.subprocess, 21
 - puffotter.test, 17
 - puffotter.test.TestCrypto, 16
 - puffotter.test.TestOs, 16
 - puffotter.test.TestUnits, 16
 - puffotter.testutils, 21
 - puffotter.units, 21

Symbols

`__init__()` (*puffotter.flask.db.ApiKey.ApiKey method*), 3
`__init__()` (*puffotter.flask.db.TelegramChatId.TelegramChatId method*), 4
`__init__()` (*puffotter.flask.db.User.User method*), 5
`__init__()` (*puffotter.flask.exceptions.ApiException method*), 15
`__init__()` (*puffotter.graphql.GraphQLClient method*), 17
`__init__()` (*puffotter.logging.ColorLogger method*), 18

A

`aggressive_request()` (*in module puffotter.requests*), 21
AlertSeverity (*class in puffotter.flask.enums*), 14
`api()` (*in module puffotter.flask.routes.decorators*), 6
`api_keys` (*puffotter.flask.db.User.User attribute*), 5
`api_login_required()` (*in module puffotter.flask.routes.decorators*), 6
`API_VERSION` (*puffotter.flask.Config.Config attribute*), 12
ApiException, 15
ApiKey (*class in puffotter.flask.db.ApiKey*), 3
`app` (*in module puffotter.flask.base*), 14
`argparse_add_logfile()` (*in module puffotter.init*), 18
`argparse_add_verbosity()` (*in module puffotter.init*), 18

B

`base_url()` (*puffotter.flask.Config.Config class method*), 14
`BEHIND_PROXY` (*puffotter.flask.Config.Config attribute*), 12
`blueprint_generators` (*in module puffotter.flask.routes*), 7
`byte_string_to_byte_count()` (*in module puffotter.units*), 21

C

`chat_id` (*puffotter.flask.db.TelegramChatId.TelegramChatId attribute*), 4
`ChatId.start()` (*in module puffotter.init*), 18
ColorLogger (*class in puffotter.logging*), 18
Config (*class in puffotter.flask.Config*), 12
`confirmation_hash` (*puffotter.flask.db.User.User attribute*), 5
`confirmed` (*puffotter.flask.db.User.User attribute*), 5
`create_file()` (*in module puffotter.os*), 19
`CREATED_BLUEPRINTS` (*in module puffotter.flask.initialize*), 15
`creation_time` (*puffotter.flask.db.ApiKey.ApiKey attribute*), 3

D

`DANGER` (*puffotter.flask.enums.AlertSeverity attribute*), 14
`db` (*in module puffotter.flask.base*), 14
`DB_MODE` (*puffotter.flask.Config.Config attribute*), 12
`DB_URI` (*puffotter.flask.Config.Config attribute*), 12
`debug()` (*puffotter.logging.ColorLogger method*), 18
`DEBUG_LOGGING_PATH` (*puffotter.flask.Config.Config attribute*), 12
`define_blueprint()` (*in module puffotter.flask.routes.api.user_management*), 6
`define_blueprint()` (*in module puffotter.flask.routes.static*), 6
`define_blueprint()` (*in module puffotter.flask.routes.user_management*), 6
`DOMAIN_NAME` (*puffotter.flask.Config.Config attribute*), 12
`dump_env_variables()` (*puffotter.flask.Config.Config class method*), 14

E

`email` (*puffotter.flask.db.User.User attribute*), 5
`ensure_environment_variables_present()` (*puffotter.flask.Config.Config class method*), 14
`environment_variables()` (*puffotter.flask.Config.Config class method*), 14
`error()` (*puffotter.logging.ColorLogger method*), 19

`execute_command()` (in module `puffotter.subprocess`), 21

F

`FLASK_SECRET` (`puffotter.flask.Config.Config` attribute), 12

G

`generate_hash()` (in module `puffotter.crypto`), 17
`generate_random()` (in module `puffotter.crypto`), 17
`get_ext()` (in module `puffotter.os`), 19
`get_id()` (`puffotter.flask.db.User.User` method), 5
`get_inbox_count()` (in module `puffotter.imap`), 17
`GraphQLClient` (class in `puffotter.graphql`), 17

H

`has_expired()` (`puffotter.flask.db.ApiKey.ApiKey` method), 3
`HTTP_PORT` (`puffotter.flask.Config.Config` attribute), 12
`human_readable_bytes()` (in module `puffotter.units`), 21

I

`id` (`puffotter.flask.db.ApiKey.ApiKey` attribute), 4
`id` (`puffotter.flask.db.ModelMixin.ModelMixin` attribute), 4
`id` (`puffotter.flask.db.TelegramChatId.TelegramChatId` attribute), 4
`id` (`puffotter.flask.db.User.User` attribute), 5
`INFO` (`puffotter.flask.enums.AlertSeverity` attribute), 14
`info()` (`puffotter.logging.ColorLogger` method), 19
`init_flask()` (in module `puffotter.flask.initialize`), 15
`initialize_telegram()` (`puffotter.flask.Config.Config` class method), 14
`is_active()` (`puffotter.flask.db.User.User` property), 5
`is_anonymous()` (`puffotter.flask.db.User.User` property), 5
`is_authenticated()` (`puffotter.flask.db.User.User` property), 5

J

`jsonify_models()` (in module `puffotter.json`), 18

K

`key_hash` (`puffotter.flask.db.ApiKey.ApiKey` attribute), 4

L

`listdir()` (in module `puffotter.os`), 19
`load_config()` (`puffotter.flask.Config.Config` class method), 14
`load_env_file()` (in module `puffotter.env`), 17

`LOGGING_PATH` (`puffotter.flask.Config.Config` attribute), 13

`login_manager` (in module `puffotter.flask.base`), 14

M

`makedirs()` (in module `puffotter.os`), 19
`MAX_API_KEY_AGE` (`puffotter.flask.Config.Config` attribute), 13
`MAX_USERNAME_LENGTH` (`puffotter.flask.Config.Config` attribute), 13
`MIN_USERNAME_LENGTH` (`puffotter.flask.Config.Config` attribute), 13
`ModelMixin` (class in `puffotter.flask.db.ModelMixin`), 4
module

- `puffotter`, 22
- `puffotter.crypto`, 17
- `puffotter.env`, 17
- `puffotter.flask`, 16
- `puffotter.flask.background`, 3
- `puffotter.flask.background.telegram`, 3
- `puffotter.flask.base`, 14
- `puffotter.flask.Config`, 12
- `puffotter.flask.db`, 6
- `puffotter.flask.db.ApiKey`, 3
- `puffotter.flask.db.ModelMixin`, 4
- `puffotter.flask.db.TelegramChatId`, 4
- `puffotter.flask.db.User`, 5
- `puffotter.flask.enums`, 14
- `puffotter.flask.exceptions`, 15
- `puffotter.flask.initialize`, 15
- `puffotter.flask.routes`, 7
- `puffotter.flask.routes.api`, 6
- `puffotter.flask.routes.api.user_management`, 6
- `puffotter.flask.routes.decorators`, 6
- `puffotter.flask.routes.static`, 6
- `puffotter.flask.routes.user_management`, 6
- `puffotter.flask.test`, 12
- `puffotter.flask.test.db`, 8
- `puffotter.flask.test.db.TestApiKey`, 7
- `puffotter.flask.test.db.TestModelMixin`, 7
- `puffotter.flask.test.db.TestUser`, 8
- `puffotter.flask.test.misc`, 10
- `puffotter.flask.test.misc.TestApiCalls`, 8
- `puffotter.flask.test.misc.TestConfig`, 9
- `puffotter.flask.test.misc.TestErrorHandling`, 9

puffotter.flask.test.misc.TestInitialization, 16
 9
 puffotter.flask.test.misc.TestWsgi, 9
 9
 puffotter.flask.test.routes, 12
 puffotter.flask.test.routes.api, 10
 puffotter.flask.test.routes.api.TestApiKeyRoute, 10
 10
 puffotter.flask.test.routes.TestForgotRoute, 10
 10
 puffotter.flask.test.routes.TestLoginRoute, 11
 11
 puffotter.flask.test.routes.TestProfileRoute, 11
 11
 puffotter.flask.test.routes.TestRegisterRoute, 11
 11
 puffotter.flask.test.routes.TestStaticRoutes, 12
 12
 puffotter.flask.test.TestFramework, 12
 12
 puffotter.flask.wsgi, 15
 puffotter.graphql, 17
 puffotter.imap, 17
 puffotter.init, 18
 puffotter.json, 18
 puffotter.logging, 18
 puffotter.os, 19
 puffotter.print, 20
 puffotter.prompt, 20
 puffotter.recaptcha, 21
 puffotter.requests, 21
 puffotter.smtp, 21
 puffotter.subprocess, 21
 puffotter.test, 17
 puffotter.test.TestCrypto, 16
 puffotter.test.TestOs, 16
 puffotter.test.TestUnits, 16
 puffotter.testutils, 21
 puffotter.units, 21

P

password_hash (*puffotter.flask.db.User.User* attribute), 5
 pprint () (*in module puffotter.print*), 20
 prompt () (*in module puffotter.prompt*), 20
 prompt_comma_list () (*in module puffotter.prompt*), 20
 puffotter
 module, 22
 puffotter.crypto
 module, 17
 puffotter.env
 module, 17
 puffotter.flask
 module, 22
 puffotter.flask.background
 module, 3
 puffotter.flask.background.telegram
 module, 3
 puffotter.flask.base
 module, 14
 puffotter.flask.Config
 module, 12
 puffotter.flask.db
 module, 6
 puffotter.flask.db.ApiKey
 module, 3
 puffotter.flask.db.ModelMixin
 module, 4
 puffotter.flask.db.TelegramChatId
 module, 4
 puffotter.flask.db.User
 module, 5
 puffotter.flask.enums
 module, 14
 puffotter.flask.exceptions
 module, 15
 puffotter.flask.initialize
 module, 15
 puffotter.flask.routes
 module, 7
 puffotter.flask.routes.api
 module, 6
 puffotter.flask.routes.api.user_management
 module, 6
 puffotter.flask.routes.decorators
 module, 6
 puffotter.flask.routes.static
 module, 6
 puffotter.flask.routes.user_management
 module, 6
 puffotter.flask.test
 module, 12
 puffotter.flask.test.db
 module, 8
 puffotter.flask.test.db.TestApiKey
 module, 7
 puffotter.flask.test.db.TestModelMixin
 module, 7
 puffotter.flask.test.db.TestUser
 module, 8
 puffotter.flask.test.misc
 module, 10
 puffotter.flask.test.misc.TestApiCalls
 module, 8
 puffotter.flask.test.misc.TestConfig
 module, 9
 puffotter.flask.test.misc.TestErrorHandling

module, 9
 puffotter.flask.test.misc.TestInitializa
 module, 9
 puffotter.flask.test.misc.TestWsgi
 module, 9
 puffotter.flask.test.routes
 module, 12
 puffotter.flask.test.routes.api
 module, 10
 puffotter.flask.test.routes.api.TestApiKeyRoute
 module, 10
 puffotter.flask.test.routes.TestForgotRoute
 module, 10
 puffotter.flask.test.routes.TestLoginRoute
 module, 11
 puffotter.flask.test.routes.TestProfileRoute
 module, 11
 puffotter.flask.test.routes.TestRegisterRoute
 module, 11
 puffotter.flask.test.routes.TestStaticRoutes
 module, 12
 puffotter.flask.test.TestFramework
 module, 12
 puffotter.flask.wsgi
 module, 15
 puffotter.graphql
 module, 17
 puffotter.imap
 module, 17
 puffotter.init
 module, 18
 puffotter.json
 module, 18
 puffotter.logging
 module, 18
 puffotter.os
 module, 19
 puffotter.print
 module, 20
 puffotter.prompt
 module, 20
 puffotter.recaptcha
 module, 21
 puffotter.requests
 module, 21
 puffotter.smtp
 module, 21
 puffotter.subprocess
 module, 21
 puffotter.test
 module, 17
 puffotter.test.TestCrypto
 module, 16
 puffotter.test.TestOs

module, 16
 puffotter.test.TestUnits
 module, 16
 puffotter.testutils
 module, 21
 puffotter.units
 module, 21

Q

query (puffotter.graphql.GraphQLClient method), 17

R

RECAPTCHA_SECRET_KEY (puffotter.flask.Config.Config attribute), 13
 RECAPTCHA_SITE_KEY (puffotter.flask.Config.Config attribute), 13
 replace_illegal_ntfs_chars() (in module puffotter.os), 19
 REQUIRED_TEMPLATES (puffotter.flask.Config.Config attribute), 13

S

selection_prompt() (in module puffotter.prompt), 20
 send_email() (in module puffotter.smtp), 21
 send_message() (puffotter.flask.db.TelegramChatId.TelegramChatId method), 4
 SENTRY_DSN (puffotter.flask.Config.Config attribute), 13
 setup_logging() (in module puffotter.init), 18
 SMTP_ADDRESS (puffotter.flask.Config.Config attribute), 13
 SMTP_HOST (puffotter.flask.Config.Config attribute), 13
 SMTP_PASSWORD (puffotter.flask.Config.Config attribute), 13
 SMTP_PORT (puffotter.flask.Config.Config attribute), 13
 start_server() (in module puffotter.flask.wsgi), 15
 STRINGS (puffotter.flask.Config.Config attribute), 13
 SUCCESS (puffotter.flask.enums.AlertSeverity attribute), 14

T

TELEGRAM_API_KEY (puffotter.flask.Config.Config attribute), 13
 TELEGRAM_BOT_CONNECTION (puffotter.flask.Config.Config attribute), 13
 telegram_chat_id (puffotter.flask.db.User.User attribute), 5
 TELEGRAM_WHOAMI (puffotter.flask.Config.Config attribute), 13
 telegram_whoami() (in module puffotter.flask.background.telegram), 3

TelegramChatId (class in puffotter:flask.db.TelegramChatId), 4

TEMPLATE_EXTRAS (puffotter:flask.Config.Config attribute), 13

test_404() (puffotter:flask.test.misc.TestErrorHandling.TestErrorHandling method), 9

test_changing_password() (puffotter:flask.test.routes.TestProfileRoute.TestProfileRoute method), 11

test_confirming() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

test_converting_bytes_to_human_readable() (puffotter:test.TestUnits.TestCrypto method), 16

test_db_config() (puffotter:flask.test.misc.TestConfig.TestConfig method), 9

test_enum_attributes() (puffotter:flask.test.db.TestModelMixin.TestModelMixin method), 7

test_equality() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_equality() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_exception() (puffotter:flask.test.misc.TestErrorHandling.TestErrorHandling method), 9

test_exception_in_api_route() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_expiration() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_expired_api_key() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_flask_properties() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_get_about() (puffotter:flask.test.routes.TestStaticRoutes.TestStaticRoutes method), 12

test_get_index() (puffotter:flask.test.routes.TestStaticRoutes.TestStaticRoutes method), 12

test_get_privacy() (puffotter:flask.test.routes.TestStaticRoutes.TestStaticRoutes method), 12

test_hashing() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_hashing() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_hashing() (puffotter:test.TestCrypto.TestCrypto method), 16

test_error_handling_strings() (puffotter:test.TestCrypto.TestCrypto method), 16

test_invalid_confirm() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

test_invalid_login_attempts() (puffotter:flask.test.routes.TestLoginRoute.TestLoginRoute method), 11

test_invalid_recaptcha() (puffotter:flask.test.routes.TestForgotRoute.TestForgotRoute method), 10

test_invalid_recaptcha() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

test_invalid_registrations() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

test_json_representation() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_json_representation() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_json_with_telegram_chat_id() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_listdir() (puffotter:test.TestOs.TestCrypto method), 16

test_logging_in_and_out() (puffotter:flask.test.routes.TestLoginRoute.TestLoginRoute method), 11

test_missing_environment_variables() (puffotter:flask.test.misc.TestInitialization.TestInitialization method), 9

test_missing_required_template() (puffotter:flask.test.misc.TestInitialization.TestInitialization method), 9

test_non_base64_header() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_page_get() (puffotter:flask.test.routes.TestForgotRoute.TestForgotRoute method), 10

test_page_get() (puffotter:flask.test.routes.TestLoginRoute.TestLoginRoute method), 11

test_page_get() (puffotter:flask.test.routes.TestProfileRoute.TestProfileRoute method), 11

test_page_get() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

method), 11

test_random_exception() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_registering_user() (puffotter:flask.test.routes.TestRegisterRoute.TestRegisterRoute method), 11

test_repr() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_repr() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_requesting_api_key() (puffotter:flask.test.routes.api.TestApiKeyRoute.TestApiKeyRoute method), 10

test_requesting_invalid_api_keys() (puffotter:flask.test.routes.api.TestApiKeyRoute.TestApiKeyRoute method), 10

test_resetting_password() (puffotter:flask.test.routes.TestForgotRoute.TestForgotRoute method), 10

test_revoking_api_key() (puffotter:flask.test.routes.api.TestApiKeyRoute.TestApiKeyRoute method), 10

test_salt() (puffotter:test.TestCrypto.TestCrypto method), 16

test_starting_and_stopping_wsgi_server() (puffotter:flask.test.misc.TestWsgi.TestWsgi method), 9

test_starting_background_tasks() (puffotter:flask.test.misc.TestWsgi.TestWsgi method), 9

test_string_representation() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_string_representation() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_unauthorized_call() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_unsuccessful_password_change() (puffotter:flask.test.routes.TestProfileRoute.TestProfileRoute method), 11

test_unsuccessful_user_delete() (puffotter:flask.test.routes.TestProfileRoute.TestProfileRoute method), 11

test_unsuccessfully_resetting_password() (puffotter:flask.test.routes.TestForgotRoute.TestForgotRoute method), 10

test_unsuccessfully_revoking_api_key() (puffotter:flask.test.routes.api.TestApiKeyRoute.TestApiKeyRoute method), 10

test_user_delete() (puffotter:flask.test.routes.TestProfileRoute.TestProfileRoute method), 11

method), 11

test_using_non_json_data() (puffotter:flask.test.misc.TestApiCalls.TestApiCalls method), 8

test_verifying_key() (puffotter:flask.test.db.TestApiKey.TestApiKey method), 7

test_verifying_password() (puffotter:flask.test.db.TestUser.TestUser method), 8

test_verifying_with_invalid_hash() (puffotter:test.TestCrypto.TestCrypto method), 16

version() (puffotter:flask.test.misc.TestConfig.TestConfig method), 9

models (class in puffotter:flask.test.misc.TestApiCalls), 8

TestApiKey (class in puffotter:flask.test.db.TestApiKey), 7

TestApiKeyRoute (class in puffotter:flask.test.routes.api.TestApiKeyRoute), 10

TestConfig (class in puffotter:flask.test.misc.TestConfig), 9

TestCrypto (class in puffotter:test.TestCrypto), 16

TestCrypto (class in puffotter:test.TestOs), 16

TestCrypto (class in puffotter:test.TestUnits), 16

TestErrorHandling (class in puffotter:flask.test.misc.TestErrorHandling), 9

TestForgotRoute (class in puffotter:flask.test.routes.TestForgotRoute), 10

TESTING (puffotter:flask.Config.Config attribute), 13

TestInitialization (class in puffotter:flask.test.misc.TestInitialization), 9

TestLoginRoute (class in puffotter:flask.test.routes.TestLoginRoute), 11

TestModelMixin (class in puffotter:flask.test.db.TestModelMixin), 7

TestProfileRoute (class in puffotter:flask.test.routes.TestProfileRoute), 11

TestRegisterRoute (class in puffotter:flask.test.routes.TestRegisterRoute), 11

TestStaticRoutes (class in puffotter:flask.test.routes.TestStaticRoutes), 12

TestUser (class in puffotter:flask.test.db.TestUser), 8

TestWsgi (class in puffotter:flask.test.misc.TestWsgi), 9

User (class in puffotter:flask.db.User), 5

user (puffotter:flask.db.ApiKey.ApiKey attribute), 4

user (puffotter:flask.db.TelegramChatId.TelegramChatId attribute), 4

user_id (puffotter:flask.db.ApiKey.ApiKey attribute), 4

`user_id` (*puffotter.flask.db.TelegramChatId.TelegramChatId attribute*), 4
`username` (*puffotter.flask.db.User.User attribute*), 5

V

`VERBOSEITY` (*puffotter.flask.Config.Config attribute*), 13
`verify_confirmation()` (*puffotter.flask.db.User.User method*), 5
`verify_key()` (*puffotter.flask.db.ApiKey.ApiKey method*), 4
`verify_password()` (*in module puffotter.crypto*), 17
`verify_password()` (*puffotter.flask.db.User.User method*), 5
`verify_recaptcha()` (*in module puffotter.recaptcha*), 21
`VERSION` (*puffotter.flask.Config.Config attribute*), 13

W

`WARNING` (*puffotter.flask.enums.AlertSeverity attribute*), 14
`warning()` (*puffotter.logging.ColorLogger method*), 19
`WARNING_LOGGING_PATH` (*puffotter.flask.Config.Config attribute*), 13

Y

`yn_prompt()` (*in module puffotter.prompt*), 20