status-page

Release 0.1.1

Mar 12, 2019

CONTENTS

Contents:

CHAPTER

ONE

STATUS_PAGE PACKAGE

1.1 Subpackages

1.1.1 status_page.analytics package

Submodules

status_page.analytics.ping module

status_page.analytics.ping.**ping**(*address: str*) → bool
Pings an address and checks whether or not the address responds :param address: The address to ping :return: Whether or not the device responds to the ping

status_page.analytics.ping.**port_ping**(*address: str*, *port: int*) → bool
Pings an address at a specific port using nmap and checks whether or not the address responds :param address: The address to ping :param port: The port to ping :return: Whether or not the device responds to the ping

status_page.analytics.version module

status_page.analytics.version.**get_gitlab_version**(*address: str*, *token: str*) → str
Gets the version of a gitlab instance :param address: The address of the gitlab instance (excluding https) :param token: The API token used for authentication :return: The version, minus any

status_page.analytics.version.**get_version_from_version_file**(*version_file: str*) → str
Retrieves the version based on a version file :param version_file: The version file to read :return: The version

status_page.analytics.web module

status_page.analytics.web.**test_webpage**(*address: str*) → bool
Checks whether or not a web page is accesible :param address: The address to check :return: True if the web page is accessible (Code 200), else False.

Module contents

status_page.analytics.**generate_analytics_data**(*server_list_config: Dict[str, Dict[str, Any]]*) → Dict[str, Dict[str, Any]]
Analyzes the servers provided as argument and generates a dictionary containing data which can be visualized. :param server_list_config: The config to analyze :return: The analyzed data

1.1.2 status_page.models package

Submodules

status_page.models.User module

class status_page.models.User.**User**(**args*, ***kwargs*)
Bases: 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

email
The user's email address

get_id() → str
Method required by flask-login :return: The user's ID as a unicode string

id
The ID is the primary key of the table and increments automatically

is_active
Property required by flask-login :return: True

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

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

password_hash
The user's hashed password, salted and hashed.

username
The user's username

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

1.1.3 status_page.routes package

Submodules

status_page.routes.routes module

Module contents

status_page.routes.**load_routes**()
Loads all application routes :return: None

1.1.4 status_page.utils package

Submodules

status_page.utils.crypto module

status_page.utils.crypto.**generate_hash**(*password: str*) → bytes
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

status_page.utils.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

status_page.utils.env module

status_page.utils.env.**load_secrets**(*secrets_file: str*)
Loads a JSON file filled with configuration details and secrets into os.environ :param secrets_file: The file to load :return: None

status_page.utils.env.**resolve_env_variable**(*env_key: str*, *_type: type = <class 'str'>*, *default: object = None*) → object
Resolves an environment key. A non-existant environment key will lead to a KeyError unless the app is in testing mode, in which case database-related variables won't cause a KeyError. KeyErrors can also be provided using the 'default' argument :param env_key: The environment key to resolve :param _type: The type of the environment variable :param default: An optional default value :return: The resolved environment variable.

None if the app is in testing mode and the variable is db-related

status_page.utils.initialize module

status_page.utils.initialize.**initialize_app**()
Initializes the App :return: None

status_page.utils.initialize.**initialize_db**(*db_uri: str*)
Initializes the SQLAlchemy database :param db_uri: The URI of the database to initialize :return: None

status_page.utils.initialize.**initialize_login_manager**()
Initializes the login manager :return: None

Module contents

1.2 Submodules

1.3 status_page.config module

1.4 status_page.run module

1.5 Module contents

status_page.**app = <Flask 'status_page'>**
The Flask App

status_page.**db = <SQLAlchemy engine=None>**
The SQLAlchemy database connection

status_page.**login_manager = <flask_login.login_manager.LoginManager object>**
The Flask-Login Login Manager

CHAPTER

TWO

STATUS_PAGE

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

A

D

E

G

I

L

P

R

S

T

U

V