Airgonimic_Backend

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 backend.resources.configuration Namespace Reference

3.1.1 Detailed Description

Author

Sebastian Steinmeyer Handles the configuration ressources. See rest api documentation for further information.

3.2 backend.resources.information Namespace Reference

3.2.1 Detailed Description

Author

Sebastian Steinmeyer Handles the information ressources. See rest api documentation for further information.

3.3 backend.resources.login Namespace Reference

3.3.1 Detailed Description

Author

Sebastian Steinmeyer Handles the login ressources. See rest api documentation for further information.

3.4 backend.resources.register Namespace Reference

3.4.1 Detailed Description

Author

Sebastian Steinmeyer Handles the register ressources. See rest api documentation for further information.

3.5 backend.resources.room Namespace Reference

3.5.1 Detailed Description

Author

Sebastian Steinmeyer Handles the room ressources. See rest api documentation for further information.

3.6 backend.resources.user Namespace Reference

3.6.1 Detailed Description

Author

Sebastian Steinmeyer Handles the user ressources. See rest api documentation for further information.

3.7 backend.resources.window Namespace Reference

3.7.1 Detailed Description

Author

Sebastian Steinmeyer Handles the window ressources. See rest api documentation for further information.

3.8 backend.util.__init__ Namespace Reference

3.8.1 Detailed Description

Author

Sebastian Steinmeyer The application factory of the backend

3.9 backend.util.arg_parser Namespace Reference

3.9.1 Detailed Description

Author

Sebastian Steinmeyer Offers the functinallity to simple parse integer and float within defined range.

3.10 backend.util.db Namespace Reference

3.10.1 Detailed Description

Author

Sebastian Steinmeyer Handles the functionality for the database access

3.11 backend.util.response_code Namespace Reference

3.11.1 Detailed Description

Author

Sebastian Steinmeyer Offers constants for the http status codes used in the backend.

3.12 backend.util.security Namespace Reference

3.12.1 Detailed Description

Author

Sebastian Steinmeyer Handles the checks for user rights and getting room/user by security token. At the moment the token is simply the id of the user/room.

3.13 control.__init__ Namespace Reference

3.13.1 Detailed Description

Author

Sebastian Steinmeyer Handles the start, close and runtime of the window control.

3.14 control.configuration_handler Namespace Reference

Functions

- def parse_file (columns)

  Read the configuration as a map with lists from file.
- def load_configuration ()

  Parse the configuration and create rooms and windows based on it.
- def save_configuration (sleep_duration, room)

  Saves the configuration into configuration file.

Variables

- string filename = 'config.txt'

  The filename of the configuration file.

3.14.1 Detailed Description

Author

Sebastian Steinmeyer Handles the load and safe of configurations from file.

3.14.2 Function Documentation

3.14.2.1 load_configuration()

```
def control.configuration_handler.load_configuration ( )
```

Parse the configuration and create rooms and windows based on it.

The values backend, window_count, sleep_duration and the window gpios in form of window_i„[gpio] need to be definded in the configuration file. In case room is not given, a new room with given windows will be created. Sample structure of the configuration file: backend,<ip:port>:int,None sleep_duration,<duration>↩ :int,None window_count,<count>:int,None room,<id in="" the="" backend>="">:int,None window_i,<is in="" the="" backend>="">:int,<gpio>:int

Returns

the parsed Room with the parsed Windows

3.14.2.2 parse_file()

```
def control.configuration_handler.parse_file (
              columns )
```

Read the configuration as a map with lists from file.

The seperator ist defined to ",".

Parameters

columns	the number of columns to read in each row.

Returns

a dictionary with a list of parsed values

3.14.2.3 save_configuration()

```
def control.configuration_handler.save_configuration (
              sleep_duration,
              room )
```

Saves the configuration into configuration file.

Parameters

sleep_duration	the defined duration
room	the room of type Room to get the configuration from

3.14.3 Variable Documentation

3.14.3.1 filename

```
string control.configuration_handler.filename = 'config.txt'
```

The filename of the configuration file.

3.15 control.led_handler Namespace Reference

Functions

- def open_window (gpio)
  Turn on a led.
- def close_window (gpio)
  Turn off a led.

3.15.1 Detailed Description

Author

Sebastian Steinmeyer Handles the functionallity to turn a led on and off.

3.15.2 Function Documentation

3.15.2.1 close_window()

```
def control.led_handler.close_window (
              gpio )
```

Turn off a led.

Parameters

gpio	the gpoi pin which should turn off.

3.15.2.2 open_window()

```
def control.led_handler.open_window (
              gpio )
```

Turn on a led.

Parameters

gpio	the gpoi pin which should turn on.

3.16 control.room Namespace Reference

Classes

- class Room

  Handles the functionality of a room.

Functions

- def register_new_room (backend)

  Registers a new room at the given backend.

3.16.1 Detailed Description

Author

Sebastian Steinmeyer Handles the class Room and a funtion to register new rooms.

3.16.2 Function Documentation

3.16.2.1 register_new_room()

```
def control.room.register_new_room (
              backend )
```

Registers a new room at the given backend.

Parameters

backend	the ip and port of the target backend as string

Returns

the created Room object or None in case of error

3.17 control.sensors.read_bme280_hum Namespace Reference

3.17.1 Detailed Description

Author

Sebastian Steinmeyer Handles the functionality to read the bme280 sensor

3.18 control.sensors.read_t6613_co2 Namespace Reference

3.18.1 Detailed Description

Author

Sebastian Steinmeyer Handles the functionality to read the t6613 sensor

3.19 control.window Namespace Reference

Classes

- class Window

  Handles the functionality of a window.

Functions

- def register_new_window (backend, room_id, gpio)

  Registers a new window at the given backend.

3.19.1 Detailed Description

Author

Sebastian Steinmeyer Handles the class Window and a funtion to register new windows.

3.19.2 Function Documentation

3.19.2.1 register_new_window()

```
def control.window.register_new_window (
              backend,
              room_id,
              gpio )
```

Registers a new window at the given backend.

Parameters

backend	the ip and port of the target backend as string
room↩ _id	the id of the window containing room
gpio	the gpio pin of the used led

Returns

the created Window object or None in case of error

Chapter 4

Class Documentation

4.1 control.room.Room Class Reference

Handles the functionality of a room.

Public Member Functions

- def __init__ (self, id, backend)

  Standart constructor to initiate a room with default values.
- def __str__ (self)

  Get the room as string representation.
- def get_id (self)

  Get the room id.
- def get_update (self)

  Updates the room with the data from the given backend.
- def set_update (self)

  Pushes the room values to the given backend.
- def **add_window** (self, window)
- def check_status (self)

  Checks the air values and decide to open the windows or not.
- def get_threshold (self)

  Get the threshold from the given backend.
- def get_configuration (self)

  Get the configuration of the room and its windows.
- def get_window_count (self)

  Get the count of containing windows.
- def close_all (self)

  Closes all containing windows.

Public Attributes

- **id**
- **backend_raw**
- **backend**
- **co2**
- **humidity**
- **automatic**
- **open**
- **force**
- **windows**
- **override**

4.1.1 Detailed Description

Handles the functionality of a room.

This include getting and setting updates and check the air values.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def control.room.Room.__init__ (
              self,
              id,
              backend )
```

Standart constructor to initiate a room with default values.

Please call get_update after creating a room.

Parameters

self	the object pointer
id	the room id in the backend
backend	the ip and port of the used backend

4.1.3 Member Function Documentation

4.1.3.1 __str__()

```
def control.room.Room.__str__ (
              self )
```

Get the room as string representation.

Parameters

self	the object pointer

Returns

the room as string

4.1.3.2 check_status()

```
def control.room.Room.check_status (
              self )
```

Checks the air values and decide to open the windows or not.

In case one value is over threshold it calls open for all windows. It contains an override, which lasts till the user change it.

Parameters

self	the object pointer

4.1.3.3 close_all()

```
def control.room.Room.close_all (
              self )
```

Closes all containing windows.

Parameters

self	the object pointer

4.1.3.4 get_configuration()

```
def control.room.Room.get_configuration (
              self )
```

Get the configuration of the room and its windows.

Parameters

self	the object pointer

Returns

the configuration as string

4.1.3.5 get_id()

```
def control.room.Room.get_id (
              self )
```

Get the room id.

Parameters

self	the object pointer

Returns

the id of the room

4.1.3.6 get_threshold()

```
def control.room.Room.get_threshold (
              self )
```

Get the threshold from the given backend.

Parameters

self	the object pointer.

Returns

a dictionary with the received values.

4.1.3.7 get_update()

```
def control.room.Room.get_update (
              self )
```

Updates the room with the data from the given backend.

Parameters

self	the object pointer.

4.1.3.8 get_window_count()

```
def control.room.Room.get_window_count (
              self )
```

Get the count of containing windows.

Parameters

self	the object pointer

Returns

the window count as integer

4.1.3.9 set_update()

```
def control.room.Room.set_update (
              self )
```

Pushes the room values to the given backend.

Parameters

self	the object pointer

The documentation for this class was generated from the following file:

- control/room.py

4.2 control.window.Window Class Reference

Handles the functionality of a window.

Public Member Functions

- def __init__ (self, id, room_id, backend, gpio)

  Standart constructor to initiate a window with default values.
- def __str__ (self)

  Get the window as string representation.
- def get_update (self)

  Updates the window with the data from the given backend.
- def set_update (self)

  Pushes the window values to the given backend.
- def change_state (self, to_state)

  Changes the window into the given state.
- def get_configuration (self, index)

  Get the configuration of the window.

Public Attributes

- **id**
- **room_id**
- **backend**
- **gpio**
- **automatic**
- **open**

4.2.1 Detailed Description

Handles the functionality of a window.

This include getting and setting updates and open and close it.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def control.window.Window.__init__ (
              self,
              id,
              room_id,
              backend,
              gpio )
```

Standart constructor to initiate a window with default values.

Please call get_update after creating a window.

Parameters

self	the object pointer
id	the window id in the backend
room_id	the id of the room which contains the window
backend	the ip and port of the used backend
gpio	the gpio pin for the led which simbolize the window

4.2.3 Member Function Documentation

4.2.3.1 __str__()

```
def control.window.Window.__str__ (
              self )
```

Get the window as string representation.

Parameters

self	the object pointer

Returns

the window as string

4.2.3.2 change_state()

```
def control.window.Window.change_state (
              self,
              to_state )
```

Changes the window into the given state.

If the state is already set or automatic is disabled, nothing will happen.

Parameters

self	the object pointer
to_state	the targeted state

4.2.3.3 get_configuration()

```
def control.window.Window.get_configuration (
              self,
              index )
```

Get the configuration of the window.

Parameters

self	the object pointer
index	the window index related to the room

Returns

the configuration as string

4.2.3.4 get_update()

```
def control.window.Window.get_update (
              self )
```

Updates the window with the data from the given backend.

Parameters

self	the object pointer.

4.2.3.5 set_update()

```
def control.window.Window.set_update (
              self )
```

Pushes the window values to the given backend.

Parameters

self	the object pointer

The documentation for this class was generated from the following file:

- control/window.py

Index