

---

**state***\_machine\_db*Documentation

**Release 0.1**

**Jonatan Dellagostin**

**Apr 13, 2017**



---

## Contents

---

<b>1</b>	<b>state_machine_db</b>	<b>1</b>
1.1	Example . . . . .	1
1.2	Installation . . . . .	1
1.3	Documentation . . . . .	1
1.4	Source Code . . . . .	2
1.5	License . . . . .	2
<b>2</b>	<b>state_machine_db package contents:</b>	<b>3</b>
2.1	state_machine_db package . . . . .	3
<b>3</b>	<b>Indices and tables</b>	<b>5</b>
	<b>Python Module Index</b>	<b>7</b>



`state_machine_db` provides the implementation of a recoverable (sqlite3) state machine

### Example

```
>>> import logging
>>> logging.basicConfig()
>>> from state_machine_db import StateMachine
>>> st = StateMachine('/tmp/db.sqlite', 'first')
>>> st.logger.setLevel('DEBUG')
>>> st.start()
>>> st.update_flag = True
```

### Installation

To install `state_machine`, simply run:

```
$ pip install state_machine_db
```

`state_machine_db` is compatible with Python 2.6+ and python 3

### Documentation

[https://state\\_machine\\_db.readthedocs.io](https://state_machine_db.readthedocs.io)

## Source Code

Feel free to fork, evaluate and contribute to this project.

Source: [https://github.com/jonDel/state\\_machine\\_db](https://github.com/jonDel/state_machine_db)

## License

GPLv3 licensed.

---

state\_machine\_db package contents:

---

### state\_machine\_db package

#### Submodules

#### state\_machine\_db.state\_machine module

This module implements a state machine that waits for flags to jump from state to state until it is finished

**class** `state_machine_db.state_machine.StateMachine` (*sm\_database\_path*, *activity\_id*)  
Bases: `threading.Thread`

Implements a totally configurable state machine

#### Parameters

- **sm\_database\_path** (*str*) – path to the sqlite database
- **activity\_id** (*str*) – identifier for the current state machine instance

**static check\_if\_thread\_alive** (*activity\_id*)  
Checks if there is a thread related to *activity\_id*

**Parameters** **activity\_id** (*str*) – identifier for the current state machine instance

**Returns** True if there is a thread, False otherwise

**static get\_sm\_alive\_threads** ()  
Get info about all running threads related to state machine

**Returns** A dictionary containing the name of each running thread and its thread object

**get\_updated\_states** ()  
This method must be implemented in the child class and return, after an update in the `update_flag`, a list of updated states

**run ()**

Initiates the thread that effectively implements the state machine. A change of state must be signaled by a flag (update, must be True) The final state must be signaled by a flag (is\_finished, must be True)

## Module contents



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**S**

`state_machine_db`, 4

`state_machine_db.state_machine`, 3



## C

`check_if_thread_alive()` (`state_machine_db.state_machine.StateMachine`  
static method), 3

## G

`get_sm_alive_threads()` (`state_machine_db.state_machine.StateMachine`  
static method), 3

`get_updated_states()` (`state_machine_db.state_machine.StateMachine`  
method), 3

## R

`run()` (`state_machine_db.state_machine.StateMachine`  
method), 3

## S

`state_machine_db` (module), 4

`state_machine_db.state_machine` (module), 3

`StateMachine` (class in `state_machine_db.state_machine`),  
3