

Generating API Documentation with Sphinx

Sphinx is the software package that we use to document our SW projects.

Why Sphinx: <http://www.sphinx-doc.org/en/master/>

AUTO GENERATION OF DOCUMENTATION AS LONG AS YOU HAVE NICELY STRUCTURED DOCSTRINGS!

- Specially created for Python, which is our project language
- **Output formats:** HTML (including Windows HTML Help), LaTeX (for printable PDF versions), ePub, Texinfo, manual pages, plain text. You can always manually modify the HTML if bugs appear in the documentation
- **Extensive cross-references:** semantic markup and automatic links for functions, classes, citations, glossary terms and similar pieces of information
- **Hierarchical structure:** easy definition of a document tree, with automatic links to siblings, parents and children
- **Automatic indices:** general index as well as a language-specific module indices
- **Code handling:** automatic highlighting using the [Pygments](#) highlighter
- **Extensions:** automatic testing of code snippets, inclusion of docstrings from Python modules (API docs), and [more](#)

Tutorials

1. syntax guide: https://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html
2. quick start (from 6:14) : <https://www.youtube.com/watch?v=qrcj7sVuvUA>

Important format to follow while writing your function docstrings:

- parameters using :param <name>: <description>
- type of the parameters :type <name>: <description>
- returns using :returns: <description>
- examples (doctest)
- seealso using .. seealso:: text
- notes using .. note:: text
- warning using .. warning:: text
- todo .. todo:: text
- do not mix tabs and space

Otherwise, sphinx would fail to recognise the docstrings and could end up with sth like:

```
query.get_sim_run_eventnames(cur, simRunSetName)
:param cur :param simRunSetName user input :return calls get_sim_run_eventname_with_ids, which returns
a list of sim run event names
```

Instead of the correct one like:

```
query.get_sim_run_eventname_with_ids(cur, simRunIdStr)
Parameters: • cur –
              • simRunIdStr – a string of comma-separated sim_run_ids. eg:'1,2,3,4'
Returns: a list of sim run event names
```

If we then look at the docstrings of the two functions, the differences are that get_sim_run_eventnames does not have a ':' following 'simRunSetName' or ': return'

```
def get_sim_run_eventnames(cur, simRunSetName):
    """
    :param cur
    :param simRunSetName user input
    :return calls get_sim_run_eventname_with_ids, which returns a list of sim run event names
    """
    simRunIdStr = get_sim_run_id_str(cur, simRunSetName)
    return get_sim_run_eventname_with_ids(cur, simRunIdStr)
```

```
def get_sim_run_eventname_with_ids(cur, simRunIdStr):
    """
    :param cur:
    :param simRunIdStr: a string of comma-separated sim_run_ids. eg:'1,2,3,4'
    :return: a list of sim run event names
    """
    # IN and the following tuple is hard to substitute with ? when the length of tuple is variable
    query = "SELECT DISTINCT eventname FROM simrun WHERE id IN ({})".format(simRunIdStr)
    try:
        cur.execute(query)
        simRunsEventNamesSelected = cur.fetchall()
    except Exception as e:
        sys.exit(e)
    # print(len(simRunsEventNamesSelected))
    return map(lambda x: x[0], simRunsEventNamesSelected)
```

Sample output

event_search 0.0.1 documentation »

Next topic
event_search module

This Page
Show Source

Quick search
Go

scripts

- event_search module
- get_event module
- get_im_vf module
- get_imdb module
- get_realizations module
- get_simrunset module
- get_station module
- query module

next | modules | index

event_search 0.0.1 documentation »

Quick search
Go

Python Module Index

g | q

g

- get_event
- get_im_vf
- get_imdb
- get_realizations
- get_simrunset
- get_station

q

- query

modules | index

event_search 0.0.1 documentation »

Previous topic
get_station module

This Page
Show Source

Quick search
Go

query module

A library that contains all necessary queries to retrieve data from the sqlite seisfinder2 database for event search

query.**add_underscore**(string)

Add an underscore to a non_empty string

Parameters: string – user input

Returns: '_' + string if string not empty else empty string

query.**check_imtype**(imType)

Parameters: imType – user input

Returns: validate user input imType and convert it to the standard form if necessary.

query.**create_name**(directory, eventName, stationName, realisation, imType, latitude, longitude)

Parameters:

- directory – /home/\$username/event_search_result
- eventName – user input
- stationName – user input
- realisation – user input. Default ""
- imType – user input. Default ""
- latitude – user input. Default ""
- longitude – user input. Default ""
- csvOut – user input

Returns: abs path to the file to be created

query.**display_events**(simRunSetName, simRunsEventNamesSelected)

print all simRunsEventNamesSelected

Parameters:

- simRunSetName
- simRunsEventNamesSelected: a list of simRunsEventName

Returns: abs path to the file to be created

previous | modules | index