aiidalab-widgets-base

unknown

May 07, 2021

CONTENTS

AiiDAlab Widgets Base (AWB) is a set of handy widgets that are used in AiiDAlab to create applications.

CHAPTER

ONE

INTRODUCTION

AiiDAlab Widgets Base (AWB) is a library of widgets that is used in AiiDAlab when building applications. The main goal of most of the widgets is to provide a user friendly interface to interact with AiiDA objects. Some other widgets are more of a general purpose with a goal to help you building powerful applications.

CHAPTER

TWO

CONTRIBUTE TO AWB

If would like to contiribute to the project you can do it in multiple ways:

- Create issues in case you found a bug or you need a feature that is not yet implemented.
- Contribute new widgets.
- Help us with fixing the existing issues.
- Extend the documentation.

CHAPTER

THREE

LIST OF WIDGETS

Here we collect the information about all the widgets available in AWB and provide a simple example of how to use them.

3.1 Dealing with codes

[issue: #185]

3.2 Dealing with computers

[issue: #186]

3.3 Dealing with external databases

[issue: #187]

3.4 Export AiiDA database

[issue: #188]

3.5 Dealing with AiiDA nodes

[issue: #189]

3.6 Dealing with AiiDA processes

[issue: #190]

3.7 Dealing with structures

[issue: #191]

CHAPTER

FOUR

AIIDALAB_WIDGETS_BASE PACKAGE

4.1 Subpackages

4.1.1 aiidalab_widgets_base.data package

Module contents

Useful functions that provide access to some data.

class aiidalab_widgets_base.data.**LigandSelectorWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_selection.Dropdown

Class to select ligands that are returned as *Atoms* object

__init__(*value=0*, *description='Select ligand'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.data'

_trait_default_generators = {}

property anchoring_atom
Return anchoring atom chemical symbol.

rotate(*align_to=(0, 0, 1)*, *remove_anchor=False*)
Rotate group in such a way that vector which was (-1,-1,-1) is alligned with align_to.

4.1.2 aiidalab_widgets_base.utils package

Module contents

Some utility functions used acrross the repository.

aiidalab_widgets_base.utils.**find_ranges**(*iterable*)
Yield range of consecutive numbers.

aiidalab_widgets_base.utils.**get_ase_from_file**(*fname*, *format=None*)
Get ASE structure object.

aiidalab_widgets_base.utils.**list_to_string_range**(*lst*, *shift=1*)
Converts a list like [0, 2, 3, 4] into a string like '1 3..5'.

Shift used when e.g. for a user interface numbering starts from 1 not from 0

aiidalab_widgets_base.utils.**predefine_settings**(*obj*, ***kwargs*)
Specify some pre-defined settings.

aiidalab_widgets_base.utils.**string_range_to_list**(*strng*, *shift=- 1*)
Converts a string like '1 3..5' into a list like [0, 2, 3, 4].

Shift used when e.g. for a user interface numbering starts from 1 not from 0

aiidalab_widgets_base.utils.**valid_arguments**(*arguments*, *valid_args*)
Check whether provided arguments are valid.

4.2 Submodules

4.3 aiidalab_widgets_base.codes module

Module to manage AiiDA codes.

class aiidalab_widgets_base.codes.**AiiDACodeSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class that allows to setup AiiDA code

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.codes'

_setup_code(*_=None*)
Setup an AiiDA code.

_trait_default_generators = {}

append_text
A trait for unicode strings.

computer
A trait type representing a Union type.

description
A trait for unicode strings.

exists()
Returns True if the code exists, returns False otherwise.

input_plugin
A trait for unicode strings.

label
A trait for unicode strings.

prepend_text
A trait for unicode strings.

remote_abs_path
A trait for unicode strings.

class aiidalab_widgets_base.codes.**CodeDropdown**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Code selection widget. Attributes:

selected_code(Unicode or Code): Trait that points to the selected Code instance. It can be set either to an AiiDA Code instance or to a code label (will automatically be replaced by the corresponding Code instance). It is linked to the 'value' trait of the *self.dropdown* widget.

codes(Dict): Trait that contains a dictionary (label => Code instance) for all codes found in the AiiDA database for the selected plugin. It is linked to the 'options' trait of the *self.dropdown* widget.

allow_hidden_codes(Bool): Trait that defines whether to show hidden codes or not.

allow_disabled_computers(Bool): Trait that defines whether to show codes on disabled computers.

__init__(*input_plugin*, *description='Select code:'*, *path_to_root='../'*, ***kwargs*)
Dropdown for Codes for one input plugin.

input_plugin (str): Input plugin of codes to show.

description (str): Description to display before the dropdown.

__module__ = 'aiidalab_widgets_base.codes'

static _full_code_label(*code*)

_get_codes()
Query the list of available codes.

_trait_default_generators = {}

_validate_selected_code

allow_disabled_computers
A boolean (True, False) trait.

allow_hidden_codes
A boolean (True, False) trait.

codes
An instance of a Python dict.

One or more traits can be passed to the constructor to validate the keys and/or values of the dict. If you need more detailed validation, you may use a custom validator method.

Changed in version 5.0: Added key_trait for validating dict keys.

Changed in version 5.0: Deprecated ambiguous `trait`, `traits` args in favor of `value_trait`, `per_key_traits`.

refresh(*_=None*)
Refresh available codes.

The job of this function is to look in AiiDA database, find available codes and put them in the dropdown attribute.

selected_code
A trait type representing a Union type.

4.4 aiidalab_widgets_base.computers module

All functionality needed to setup a computer.

class aiidalab_widgets_base.computers.**AiidaComputerSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Inform AiiDA about a computer.

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.computers'

_configure_computer()
Create AuthInfo.

_on_setup_computer(*_=None*)
When setup computer button is pressed.

_trait_default_generators = {}

_validate_mpiprocs_per_machine

_validate_safe_interval

append_text
A trait for unicode strings.

description
A trait for unicode strings.

hostname
A trait for unicode strings.

label
A trait for unicode strings.

mpiprocs_per_machine
A trait type representing a Union type.

mpirun_command
A trait for unicode strings.

prepend_text
A trait for unicode strings.

safe_interval
A trait type representing a Union type.

scheduler
A trait for unicode strings.

test(*_=None*)

transport
A trait for unicode strings.

work_dir
A trait for unicode strings.

class aiidalab_widgets_base.computers.**ComputerDropdown**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select a configured computer.

Attributes:

selected_computer(Unicode or Computer): Trait that points to the selected Computer instance.
It can be set either to an AiiDA Computer instance or to a computer label (will automatically be replaced by the corresponding Computer instance). It is linked to the 'value' trait of *self._dropdown* widget.

computers(Dict): Trait that contains a dictionary (label => Computer instance) for all computers found in the AiiDA database. It is linked to the 'options' trait of *self._dropdown* widget.

allow_select_disabled(Bool): Trait that defines whether to show disabled computers.

__init__(*description='Select computer:'*, *path_to_root='../'*, ***kwargs*)
Dropdown for configured AiiDA Computers.

description (str): Text to display before dropdown.

path_to_root (str): Path to the app's root folder.

__module__ = 'aiidalab_widgets_base.computers'

_get_computers()
Get the list of available computers.

_trait_default_generators = {}

_validate_selected_computer

allow_select_disabled
A boolean (True, False) trait.

computers
An instance of a Python dict.

One or more traits can be passed to the constructor to validate the keys and/or values of the dict. If you need more detailed validation, you may use a custom validator method.

Changed in version 5.0: Added key_trait for validating dict keys.

Changed in version 5.0: Deprecated ambiguous `trait`, `traits` args in favor of `value_trait`, `per_key_traits`.

refresh(*_=None*)
Refresh the list of configured computers.

selected_computer
A trait type representing a Union type.

class aiidalab_widgets_base.computers.**SshComputerSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Setup password-free access to a computer.

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.computers'

property __password
Returning the password and immediately destroying it

property __private_key
unwrap private key file and setting filename and file content

property __proxy_password
Returning the password and immediately destroying it

static _add_private_key(*private_key_fname*, *private_key_content*)
param private_key_fname: string param private_key_content: bytes

_configure_proxy(*password*, *proxy_password*)
Configure proxy server.

_make_host_known(*hostname*, *proxycmd=None*)
Add host information into known_hosts file.

_observe_proxy_hostname

_observe_proxy_username

_on_setup_ssh(*mode*, *change*)
ATTENTION: modifying the order of operations in this function can lead to unexpected problems

static _send_pubkey(*hostname*, *username*, *password*, *proxycmd=''*)
Send a publick key to a remote host.

static _ssh_keygen()
Generate ssh key pair.

_trait_default_generators = {}

_validate_port

_write_ssh_config(*proxycmd=''*, *private_key_abs_fname=None*)
Put host information into the config file.

can_login(*silent=False*)
Check if it is possible to login into the remote host.

hostname
A trait for unicode strings.

is_host_known(*hostname=None*)
Check if the host is known already.

is_in_config()
Check if the config file contains host information.

on_setup_ssh(*change*)
Setup ssh, password and private key are supported

on_use_diff_proxy_username_change(*change*)
If using different username for proxy check-box is clicked.

on_use_proxy_change(*change*)
If proxy check-box is clicked.

on_use_verification_mode_change(*change*)
which verification mode is chosen.

port
A trait type representing a Union type.

proxy_hostname
A trait for unicode strings.

proxy_username
A trait for unicode strings.

setup_counter
An int trait.

use_proxy
A boolean (True, False) trait.

username
A trait for unicode strings.

4.5 aiidalab_widgets_base.databases module

Widgets that allow to query online databases.

class aiidalab_widgets_base.databases.**CodQueryWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Query structures in Crystallography Open Database (COD) Useful class members: :ivar structure(Atoms): trait that contains the selected structure, None if structure is not selected.

__init__(*title=''*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_default_structure

_on_click_query(*change*)
Call query when the corresponding button is pressed.

_on_select_structure(*change*)
When a structure was selected.

static _query(*idn=None*, *formula=None*)
Make the actual query.

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.databases.**CodeDatabaseWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.HBox

Extract the setup of a known computer from the AiiDA code registry.

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_trait_default_generators = {}

append_text
A trait for unicode strings.

computer
A trait for unicode strings.

description
A trait for unicode strings.

input_plugin
A trait for unicode strings.

label
A trait for unicode strings.

on_computer
A boolean (True, False) trait.

prepend_text
A trait for unicode strings.

remote_abs_path
A trait for unicode strings.

update(_=*None*)

update_codes(_=*None*)
Read settings from the YAML files and populate self.database with them.

update_computers(_=*None*)

update_settings(_=*None*)
Update code settings.

class aiidalab_widgets_base.databases.**ComputerDatabaseWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.HBox

Extract the setup of a known computer from the AiiDA code registry.

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_observe_proxy_command

_trait_default_generators = {}

allow_agent
A boolean (True, False) trait.

append_text
A trait for unicode strings.

description
A trait for unicode strings.

hostname
A trait for unicode strings.

label
A trait for unicode strings.

mpiprocs_per_machine
An int trait.

mpirun_command
A trait for unicode strings.

num_cores_per_mpiproc
An int trait.

port
An int trait.

prepend_text
A trait for unicode strings.

proxy_command
A trait for unicode strings.

proxy_hostname
A trait for unicode strings.

proxy_username
A trait for unicode strings.

queue_name
A trait for unicode strings.

safe_interval
A float trait.

scheduler
A trait for unicode strings.

shebang
A trait for unicode strings.

transport
A trait for unicode strings.

update(*_=None*)

update_computers(*_=None*)

update_settings(*_=None*)
Read settings from the YAML files and populate self.database with them.

use_login_shell
A boolean (True, False) trait.

work_dir
A trait for unicode strings.

class aiidalab_widgets_base.databases.**OptimadeQueryWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

AiiDAlab-specific OPTIMADE Query widget

Useful as a widget to integrate with the *aiidalab_widgets_base.structures.StructureManagerWidget*, embedded into applications.

NOTE: *embedded* for *OptimadeQueryFilterWidget* was introduced in *optimade-client* version 2020.11.5.

Parameters

- **embedded** (*bool*) – Whether or not to show extra database and provider information. When set to *True*, the extra information will be hidden, this is useful in situations where the widget is used in a Tab or similar, e.g., for the *aiidalab_widgets_base.structures.StructureManagerWidget*.
- **title** (*str*) – Title used for Tab header if employed in *aiidalab_widgets_base.structures.StructureManagerWidget*.

__init__(*embedded: bool = True*, *title: Optional[str] = None*, ***kwargs*) → None
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_trait_default_generators = {}

_update_structure(*change: dict*) → None
New structure chosen

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

4.6 aiidalab_widgets_base.dicts module

4.7 aiidalab_widgets_base.export module

Widgets to manage AiiDA export.

class aiidalab_widgets_base.export.**ExportButtonWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_button.Button

Export Node button.

__init__(*process*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.export'

_trait_default_generators = {}

export_aiida_subgraph(*change=None*)
Perform export when the button is pressed

4.8 aiidalab_widgets_base.misc module

Some useful classes used acrross the repository.

class aiidalab_widgets_base.misc.**CopyToClipboardButton**(***kwargs*)
Bases: ipywidgets.widgets.widget_button.Button

Button to copy text to clipboard.

__init__(**args*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.misc'

_trait_default_generators = {}

copy_to_clipboard(*change=None*)
Copy text to clipboard.

value
A trait for unicode strings.

class aiidalab_widgets_base.misc.**ReversePolishNotation**(*operators*, *additional_operands=None*)
Bases: object

Class defining operations for RPN conversion

__dict__ = mappingproxy({'__module__': 'aiidalab_widgets_base.misc', '__doc__': 'Cla

__init__(*operators*, *additional_operands=None*)
Initialize self. See help(type(self)) for accurate signature.

__module__ = 'aiidalab_widgets_base.misc'

__weakref__
list of weak references to the object (if defined)

convert(*expr*)
Convert expression to postfix.

execute(*expression*)
Execute the provided expression.

haslessorequalpriority(*opa*, *opb*)
Priority of the different operators

is_operator(*opx*)
Identifies operators

static iscloseparenthesis(*operator*)
Identifies closed paretheses.

static isopenparenthesis(*operator*)
Identifies open paretheses.

static parse_infix_notation(*condition*)
Convert a string containing the expression into a list of operators and operands.

4.9 aiidalab_widgets_base.nodes module

Widgets to work with AiiDA nodes.

class aiidalab_widgets_base.nodes.**AiidaNodeTreeNode**(***kwargs*)
Bases: ipytree.tree.Node

__init__(*pk*, *name*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.nodes'

_default_openend

_trait_default_generators = {'opened': <traitlets.traitlets.DefaultHandler object>}

class aiidalab_widgets_base.nodes.**AiidaOutputsTreeNode**(***kwargs*)
Bases: ipytree.tree.Node

__init__(*name*, *parent_pk*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

disabled
A boolean (True, False) trait.

icon
A trait for unicode strings.

class aiidalab_widgets_base.nodes.**AiidaProcessNodeTreeNode**(***kwargs*)
Bases: *aiidalab_widgets_base.nodes.AiidaNodeTreeNode*

__init__(*pk*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

class aiidalab_widgets_base.nodes.**CalcFunctionTreeNode**(***kwargs*)
Bases: *aiidalab_widgets_base.nodes.AiidaProcessNodeTreeNode*

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

icon
A trait for unicode strings.

class aiidalab_widgets_base.nodes.**CalcJobTreeNode**(***kwargs*)
Bases: *aiidalab_widgets_base.nodes.AiidaProcessNodeTreeNode*

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

icon
A trait for unicode strings.

class aiidalab_widgets_base.nodes.**NodesTreeWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_output.Output

A tree widget for the structured representation of a nodes graph.

NODE_TYPE = {<class 'aiida.orm.nodes.process.workflow.workchain.WorkChainNode'>: <cla

PROCESS_STATE_STYLE = {<ProcessState.EXCEPTED: 'excepted'>: 'danger', <ProcessState.F

PROCESS_STATE_STYLE_DEFAULT = 'default'

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.nodes'

classmethod _build_tree(*root*)
Recursively build a tree nodes graph for a given tree node.

_convert_to_tree_nodes(*old_nodes*, *new_nodes*)
Convert nodes into tree nodes while re-using already converted nodes.

classmethod _find_called(*root*)

classmethod _find_children(*root*)
Find all children of the provided AiiDA node.

classmethod _find_outputs(*root*)

_observe_nodes

_observe_tree_selected_nodes(*change*)

_refresh_output()

classmethod _to_tree_node(*node*, *name=None*, ***kwargs*)
Convert an AiiDA node to a tree node.

_trait_default_generators = {}

_update_tree_node(*tree_node*)

classmethod _walk_tree(*root*)
Breadth-first search of the node tree.

find_node(*pk*)

nodes
An instance of a Python tuple.

selected_nodes
An instance of a Python tuple.

update(_=*None*)
Refresh nodes based on the latest state of the root process and its children.

class aiidalab_widgets_base.nodes.**UnknownTypeTreeNode**(**kwargs*)
Bases: *aiidalab_widgets_base.nodes.AiidaNodeTreeNode*

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

icon
A trait for unicode strings.

class aiidalab_widgets_base.nodes.**WorkChainProcessTreeNode**(**kwargs*)
Bases: *aiidalab_widgets_base.nodes.AiidaProcessNodeTreeNode*

__module__ = 'aiidalab_widgets_base.nodes'

_trait_default_generators = {}

icon
A trait for unicode strings.

4.10 aiidalab_widgets_base.process module

Widgets to work with processes.

class aiidalab_widgets_base.process.**CalcJobOutputWidget**(**kwargs*)
Bases: ipywidgets.widgets.widget_string.Textarea

Output of a calculation.

__init__(**kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_change_calculation

_trait_default_generators = {}

calculation
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the displayed output and scroll to its end.

NOTE: when this widgets is called by ProcessFollowerWidget in non-blocking manner the auto-scrolling won't work. There used to be a function for the Textarea widget, but it didn't work properly and got removed. For more information please visit: https://github.com/jupyter-widgets/ipywidgets/issues/1815

class aiidalab_widgets_base.process.**ProcessCallStackWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_string.HTML

Widget that shows process call stack.

__init__(*title='Process Call Stack'*, *path_to_root='../'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

calc_info(*node*)
Return a string with the summary of the state of a CalculationNode.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the call stack that is shown.

class aiidalab_widgets_base.process.**ProcessFollowerWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A Widget that follows a process until finished.

__init__(*process=None*, *followers=None*, *update_interval=0.1*, *path_to_root='../'*, ***kwargs*)
Initiate all the followers.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

follow(*detach=False*)
Initiate following the process with or without blocking.

on_completed(*function*)
Run functions after a process has been completed.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()

class aiidalab_widgets_base.process.**ProcessInputsWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select and show process inputs.

__init__(*process=None*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

show_selected_input(*change=None*)
Function that displays process inputs selected in the *inputs* Dropdown widget.

class aiidalab_widgets_base.process.**ProcessListWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

List of AiiDA processes.

past_days (int): Sumulations that were submitted in the last *past_days*.

incoming_node (int, str, Node): Trait that takes node id or uuid and returns the node that must be among the input nodes of the process of interest.

outgoing_node (int, str, Node): Trait that takes node id or uuid and returns the node that must be among the output nodes of the process of interest.

process_states (list): List of allowed process states.

process_label (str): Show process states of type *process_label*.

description_contains (str): string that should be present in the description of a process node.

__init__(*path_to_root='../'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_default_process_label

_follow(*update_interval*)

_trait_default_generators = {'process_label': <traitlets.traitlets.DefaultHandler obj

_validate_incoming_node

_validate_outgoing_node

_validate_process_label

description_contains
A trait for unicode strings.

incoming_node
A trait type representing a Union type.

outgoing_node
A trait type representing a Union type.

past_days
An int trait.

process_label
A trait for unicode strings.

process_states
An instance of a Python list.

start_autoupdate(*update_interval=10*)

update(*_=None*)
Perform the query.

class aiidalab_widgets_base.process.**ProcessMonitor**(***kwargs*)
Bases: traitlets.traitlets.HasTraits

Monitor a process and execute callback functions at specified intervals.

__init__(*callbacks=None*, *on_sealed=None*, *timeout=None*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

__module__ = 'aiidalab_widgets_base.process'

_monitor_process(*process_id*)

_observe_process

_trait_default_generators = {}

join()

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.process.**ProcessNodesTreeWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A tree widget for the structured representation of a process graph.

__init__(*title='Process Tree'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_observe_process

_observe_tree_selected_nodes(*change*)

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

selected_nodes
An instance of a Python tuple.

update(*_=None*)

class aiidalab_widgets_base.process.**ProcessOutputsWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select and show process outputs.

__init__(*process=None*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

show_selected_output(*change=None*)
Function that displays process output selected in the *outputs* Dropdown widget.

class aiidalab_widgets_base.process.**ProcessReportWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_string.HTML

Widget that shows process report.

__init__(*title='Process Report'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update report that is shown.

class aiidalab_widgets_base.process.**ProgressBarWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A bar showing the proggress of a process.

__init__(*title='Progress Bar'*, ***kwargs*)
Initialize ProgressBarWidget.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

property current_state

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the bar.

class aiidalab_widgets_base.process.**RunningCalcJobOutputWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Show an output of selected running child calculation.

__init__(*title='Running Job Output'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the displayed output.

class aiidalab_widgets_base.process.**SubmitButtonWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Submit button class that creates submit button jupyter widget.

__init__(*process_class*, *inputs_generator=None*, *input_dictionary_function=None*, *description='Submit'*, *disable_after_submit=True*, *append_output=False*, ***kwargs*)
Submit Button widget.

process_class (Process): Process class to submit.

inputs_generator (func): Function that returns inputs dictionary or inputs builder.

input_dictionary_function (DEPRECATED): Function that generates input parameters dictionary.

description (str): Description written on the submission button.

disable_after_submit (bool): Whether to disable the button after the process was submitted.

append_output (bool): Whether to clear widget output for each subsequent submission.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

on_btn_submit_press(*_=None*)
When submit button is pressed.

on_click(*function*)

on_submitted(*function*)
Run functions after a process has been submitted successfully.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

aiidalab_widgets_base.process.**get_running_calcs**(*process*)
Takes a process and yeilds running children calculations.

4.11 aiidalab_widgets_base.structures module

Module to provide functionality to import structures.

class aiidalab_widgets_base.structures.**BasicStructureEditor**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget that allows for the basic structure editing.

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_trait_default_generators = {}

property action_vector
Define the action vector.

add(*_=None*)
Add atoms.

align(*_=None*)
Rotate atoms to align action vector with XYZ vector.

camera_orientation
An instance of a Python list.

copy_sel(*_=None*)
Copy selected atoms and shift by 1.0 A along X-axis.

def_axis_p1(*_=None*)
Define the first point of axis.

def_axis_p2(*_=None*)
Define the second point of axis.

def_perpendicular_to_screen(*_=None*)
Define a normalized vector perpendicular to the screen.

def_point(*_=None*)
Define the action point.

mirror(*_=None*, *norm=None*, *point=None*)
Mirror atoms on the plane perpendicular to the action vector.

mirror_3p(*_=None*)
Mirror atoms on the plane containing action vector and action point.

mod_element(*_=None*)
Modify selected atoms into the given element.

remove(*_*)
Remove selected atoms.

rotate(*_=None*)
Rotate atoms around selected point in space and vector.

sel2com()
Get center of mass of the selection.

selection
An instance of a Python list.

str2vec(*string*)

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

translate_dr(*_=None*)
Translate by dr along the selected vector.

translate_dxdydz(*_=None*)
Translate by the selected XYZ delta.

translate_to_xyz(*_=None*)
Translate to the selected XYZ position.

vec2str(*vector*)

class aiidalab_widgets_base.structures.**SmilesWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Conver SMILES into 3D structure.

SPINNER = ''

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_default_structure

_on_button_pressed(*change*)
Convert SMILES to ase structure when button is pressed.

_pybel_opt(*smile*, *steps*)
Optimize a molecule using force field and pybel (needed for complex SMILES).

_rdkit_opt(*smile*, *steps*)
Optimize a molecule using force field and rdkit (needed for complex SMILES).

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

make_ase(*species*, *positions*)
Create ase Atoms object.

mol_from_smiles(*smile*, *steps=10000*)
Convert SMILES to ase structure try rdkit then pybel

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.structures.**StructureBrowserWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to query for structures stored in the AiiDA database.

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_on_select_structure(*_=None*)

_trait_default_generators = {}

preprocess()
Search structures in AiiDA database and add formula extra to them.

search(*_=None*)
Launch the search of structures in AiiDA database.

structure
A trait type representing a Union type.

class aiidalab_widgets_base.structures.**StructureExamplesWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to provide example structures for selection.

__init__(*examples*, *title=''*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_default_structure

_on_select_structure(*change*)
When structure is selected.

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

static get_example_structures(*examples*)
Get the list of example structures.

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.structures.**StructureManagerWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Upload a structure and store it in AiiDA database.

Attributes: structure(Atoms): trait that contains the selected structure. 'None' if no structure is selected. structure_node(StructureData, CifData): trait that contains AiiDA structure object node_class(str): trait that contains structure_node type (as string).

SUPPORTED_DATA_FORMATS = {'CifData': 'cif', 'StructureData': 'structure'}

__init__(*importers*, *viewer=None*, *editors=None*, *storable=True*, *node_class=None*, ***kwargs*)

Arguments:

importers(list): list of tuples each containing the displayed name of importer and the importer object. Each object should contain 'structure' trait pointing to the imported structure. The trait will be linked to 'structure' trait of this class.

storable(bool): Whether to provide Store button (together with Store format)

node_class(str): AiiDA node class for storing the structure. Possible values: 'StructureData', 'CifData' or None (let the user decide). Note: If your workflows require a specific node class, better fix it here.

__module__ = 'aiidalab_widgets_base.structures'

_change_structure_node

_convert_to_structure_node(*structure*)
Convert structure of any type to the StructureNode object.

_default_node_class

_observe_input_structure

_observe_structure_node

_structure_changed

_structure_editors(*editors*)
Preparing structure editors.

_structure_importers(*importers*)
Preparing structure importers.

_sync_structure_node()
Synchronize the structure_node trait using the currently provided info.

_trait_default_generators = {}

input_structure
A trait type representing a Union type.

node_class
A trait for unicode strings.

store_structure(*_=None*)
Stores the structure in AiiDA database.

structure
A trait type representing a Union type.

structure_node
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

undo(*_*)
Undo modifications.

class aiidalab_widgets_base.structures.**StructureUploadWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class that allows to upload structures from user's computer.

__init__(*title=''*, *description='Upload Structure'*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_on_file_upload(*change=None*)
When file upload button is pressed.

_trait_default_generators = {}

_validate_and_fix_ase_cell(*ase_structure*, *vacuum_ang=10.0*)
Checks if the ase Atoms object has a cell set, otherwise sets it to bounding box plus specified "vacuum" space

structure
A trait type representing a Union type.

4.12 aiidalab_widgets_base.structures_multi module

Module to deal with files containing multiple structures.

class aiidalab_widgets_base.structures_multi.**MultiStructureUploadWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to deal with archives (tar or zip) containing multiple structures.

__init__(*text='Upload Zip or Tar archive'*, *node_class=None*, ***kwargs*)
Upload multiple structures and store them in AiiDA database.

Parameters

- **text** (*str*) – Text to display before upload button
- **node_class** – AiiDA node class for storing the structure. Possible values: 'StructureData', 'CifData' or None (let the user decide). Note: If your workflows require a specific node class, better fix it here.

__module__ = 'aiidalab_widgets_base.structures_multi'

_on_click_store_all(*change*)
Store all the uploaded structures.

_on_click_store_selected(*change*)

_on_file_upload(*change*)
Process the archive once it is uplodaded.

_trait_default_generators = {}

change_structure()

get_ase(*filepath*)
Get an ase object containing the structure.

static get_description(*structure_ase*, *filepath*)
Get the structure description automatically.

property node_class

refresh_view()
Refresh the structure view.

select_structure(*filepath*)
Perform structure selection.

store_structure(*filepath*, *description=None*)
Store the structure in the AiiDA database.

4.13 aiidalab_widgets_base.viewers module

Jupyter viewers for AiiDA data objects.

class aiidalab_widgets_base.viewers.**BandsDataViewer**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Viewer class for BandsData object.

Parameters **bands** (*BandsData*) – BandsData object to be viewed

__init__(*bands*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.viewers'

_trait_default_generators = {}

class aiidalab_widgets_base.viewers.**DictViewer**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

__init__(*parameter*, *downloadable=True*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.viewers'

_trait_default_generators = {}

value
Viewer class for Dict object.

Parameters

- **parameter** (*Dict*) – Dict object to be viewed
- **downloadable** (*bool*) – If True, add link/button to download the content of the object

class aiidalab_widgets_base.viewers.**FolderDataViewer**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Viewer class for FolderData object.

Parameters

- **folder** (*FolderData*) – FolderData object to be viewed
- **downloadable** (*bool*) – If True, add link/button to download the content of the selected file in the folder

__init__(*folder*, *downloadable=True*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.viewers'

_trait_default_generators = {}

change_file_view(*change=None*)

download(*change=None*)
Prepare for downloading.

class aiidalab_widgets_base.viewers.**StructureDataViewer**(***kwargs*)
Bases: *aiidalab_widgets_base.viewers._StructureDataBaseViewer*

Viewer class for AiiDA structure objects.

Attributes: structure (Atoms, StructureData, CifData): Trait that contains a structure object, which was initially provided to the viewer. It can be either directly set to an ASE Atoms object or to AiiDA structure object containing *get_ase()* method.

displayed_structure (Atoms): Trait that contains a structure object that is currently displayed (super cell, for example). The trait is generated automatically and can't be set outside of the class.

__init__(*structure=None*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.viewers'

_observe_selection_2

_observe_selection_adv

_trait_default_generators = {}

_update_displayed_structure

_update_structure_viewer

_valid_structure

create_selection_info()
Create information to be displayed with selected atoms

d_from(*operand*)

displayed_structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

name_operator(*operand*)
Defining the name operator which will handle atom kind names.

not_operator(*operand*)
Reverting the selected atoms.

parse_advanced_sel(*condition=None*)
Apply advanced selection specified in the text field.

repeat

structure
A trait type representing a Union type.

class aiidalab_widgets_base.viewers.**_StructureDataBaseViewer**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Base viewer class for AiiDA structure or trajectory objects.

Parameters **configure_view** (*bool*) – If True, add configuration tabs

DEFAULT_SELECTION_COLOR = 'green'

DEFAULT_SELECTION_OPACITY = 0.2

DEFAULT_SELECTION_RADIUS = 6

__init__(*configure_view=True*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.viewers'

_appearance_tab()
Defining the appearance tab.

_default_selection

_default_supercell

static _download(*payload*, *filename*)
Download payload as a file named as filename.

_download_tab()
Defining the download tab.

_observe_selection

_on_atom_click(*_=None*)
Update selection when clicked on atom.

_prepare_payload(*file_format=None*)
Prepare binary information.

_render_structure(*change=None*)
Render the structure with POVRAY.

_selection_tab()
Defining the selection tab.

_trait_default_generators = {'selection': <traitlets.traitlets.DefaultHandler object>

_validate_selection

apply_selection(*_=None*)
Apply selection specified in the text field.

download(*change=None*)
Prepare a structure for downloading.

highlight_atoms(*vis_list*, *color='green'*, *size=6*, *opacity=0.2*)
Highlighting atoms according to the provided list.

selection
An instance of a Python list.

selection_adv
A trait for unicode strings.

supercell
An instance of a Python list.

property thumbnail

aiidalab_widgets_base.viewers.**register_viewer_widget**(*key*)
Register widget as a viewer for the given key.

aiidalab_widgets_base.viewers.**viewer**(*obj*, *downloadable=True*, ***kwargs*)
Display AiiDA data types in Jupyter notebooks.

Parameters **downloadable** (*bool*) – If True, add link/button to download the content of displayed AiiDA object.

Returns the object itself if the viewer wasn't found.

4.14 aiidalab_widgets_base.wizard module

The wizard application allows the implication of a Wizard-like GUI.

Authors:

- Carl Simon Adorf <simon.adorf@epfl.ch>

class aiidalab_widgets_base.wizard.**WizardAppWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

ICONS = {<State.INIT: 0>: '', <State.READY: 2>: '', <State.CONFIGURED: 1>: '', <

ICON_SEPARATOR = '\u2000'

__init__(*steps*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.wizard'

_consider_auto_advance(*_=None*)
Determine whether the app should automatically advance to the next step.

This is performed whenever the current step is within the SUCCESS state and has the auto_advance attribute set to True.

_observe_selected_index

_on_click_back_button(*_*)

_on_click_next_button(*_*)

_on_click_reset_button(*_*)

_trait_default_generators = {}

_update_buttons()

_update_step_state(*_*)

_update_titles()

can_reset()

classmethod icons()
Return the icon set and return animated icons based on the current time stamp.

reset(*step=0*)
Reset the app up to the given step.

For example, with step=0 (the default), the whole app is reset. With step=1, all but the first step are reset.

selected_index
An int trait.

class aiidalab_widgets_base.wizard.**WizardAppWidgetStep**(***kwargs*)
Bases: traitlets.traitlets.HasTraits

One step of a WizardAppWidget.

class State(*value*)
Bases: enum.Enum

Each step is always in one specific state.

The state is used to determine:

1) how the step is visually presented to the user, and

2) whether the next step is accessible (i.e. reached the SUCCESS state).

App developers are encouraged to use the step states to couple application logic and interface. In general, all widget changes should trigger a re-evaluation of the step state, and states also determine whether certain widgets are enabled or disabled.

A step can be in one of the following states:

INIT: The initial state, usually all widgets disabled. READY: The step (widget) is ready for user input (some or all widgets enabled). CONFIGURED: The step is in a consistent configuration awaiting confirmation. ACTIVE: The step is carrying out a runtime operation. SUCCESS: A configuration has been confirmed / a runtime operation successfully finished. FAIL: A runtime operation has failed in an unrecoverable way.

Not all steps must implement all states, for example:

- the first step does not need an INIT state
- a step without runtime process should not have an ACTIVE or FAIL state
- a "review & confirm" step does not require a READY state.
- a step without configuration options (e.g. pure "review & confirm" step)

Important: The next step is only accessible if the current step is within the SUCCESS state!

ACTIVE = 3

CONFIGURED = 1

FAIL = -1

INIT = 0

READY = 2

SUCCESS = 4

__module__ = 'aiidalab_widgets_base.wizard'

__module__ = 'aiidalab_widgets_base.wizard'

_trait_default_generators = {}

auto_advance
A boolean (True, False) trait.

can_reset()

state
Use a Enum class as model for the data type description. Note that if no default-value is provided, the first enum-value is used as default-value.

```
# -- SINCE: Python 3.4 (or install backport: pip install enum34)
import enum
from traitlets import HasTraits, UseEnum

class Color(enum.Enum):
    red = 1         # -- IMPLICIT: default_value
    blue = 2
    green = 3

class MyEntity(HasTraits):
    color = UseEnum(Color, default_value=Color.blue)

entity = MyEntity(color=Color.red)
entity.color = Color.green    # USE: Enum-value (preferred)
entity.color = "green"        # USE: name (as string)
entity.color = "Color.green"  # USE: scoped-name (as string)
entity.color = 3              # USE: number (as int)
assert entity.color is Color.green
```

4.15 Module contents

Reusable widgets for AiiDAlab applications.

class aiidalab_widgets_base.**AiiDACodeSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class that allows to setup AiiDA code

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.codes'

_setup_code(*_=None*)
Setup an AiiDA code.

_trait_default_generators = {}

append_text
A trait for unicode strings.

computer
A trait type representing a Union type.

description
A trait for unicode strings.

exists()
Returns True if the code exists, returns False otherwise.

input_plugin
A trait for unicode strings.

label
A trait for unicode strings.

prepend_text
A trait for unicode strings.

remote_abs_path
A trait for unicode strings.

class aiidalab_widgets_base.**AiidaComputerSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Inform AiiDA about a computer.

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.computers'

_configure_computer()
Create AuthInfo.

_on_setup_computer(*_=None*)
When setup computer button is pressed.

_trait_default_generators = {}

_validate_mpiprocs_per_machine

_validate_safe_interval

append_text
A trait for unicode strings.

description
A trait for unicode strings.

hostname
A trait for unicode strings.

label
A trait for unicode strings.

mpiprocs_per_machine
A trait type representing a Union type.

mpirun_command
A trait for unicode strings.

prepend_text
A trait for unicode strings.

safe_interval
A trait type representing a Union type.

scheduler
A trait for unicode strings.

test(_=*None*)

transport
A trait for unicode strings.

work_dir
A trait for unicode strings.

class aiidalab_widgets_base.**BasicStructureEditor**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget that allows for the basic structure editing.

__annotations__ = {}

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_trait_default_generators = {}

property action_vector
Define the action vector.

add(_=*None*)
Add atoms.

align(_=*None*)
Rotate atoms to align action vector with XYZ vector.

camera_orientation
An instance of a Python list.

copy_sel(_=*None*)
Copy selected atoms and shift by 1.0 A along X-axis.

def_axis_p1(_=*None*)
Define the first point of axis.

def_axis_p2(_=*None*)
Define the second point of axis.

def_perpendicular_to_screen(_=*None*)
Define a normalized vector perpendicular to the screen.

def_point(_=*None*)
Define the action point.

mirror(_=*None*, *norm=None*, *point=None*)
Mirror atoms on the plane perpendicular to the action vector.

mirror_3p(_=*None*)
Mirror atoms on the plane containing action vector and action point.

mod_element(_=*None*)
Modify selected atoms into the given element.

remove(_)
Remove selected atoms.

rotate(_=*None*)
Rotate atoms around selected point in space and vector.

sel2com()
Get center of mass of the selection.

selection
An instance of a Python list.

str2vec(*string*)

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

translate_dr(_=*None*)
Translate by dr along the selected vector.

translate_dxdydz(_=*None*)
Translate by the selected XYZ delta.

translate_to_xyz(_=*None*)
Translate to the selected XYZ position.

vec2str(*vector*)

class aiidalab_widgets_base.**CodQueryWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Query structures in Crystallography Open Database (COD) Useful class members: :ivar structure(Atoms): trait that contains the selected structure, None if structure is not selected.

__annotations__ = {}

__init__(*title=''*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_default_structure

_on_click_query(*change*)
Call query when the corresponding button is pressed.

_on_select_structure(*change*)
When a structure was selected.

static _query(*idn=None*, *formula=None*)
Make the actual query.

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**CodeDatabaseWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.HBox

Extract the setup of a known computer from the AiiDA code registry.

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_trait_default_generators = {}

append_text
A trait for unicode strings.

computer
A trait for unicode strings.

description
A trait for unicode strings.

input_plugin
A trait for unicode strings.

label
A trait for unicode strings.

on_computer
A boolean (True, False) trait.

prepend_text
A trait for unicode strings.

remote_abs_path
A trait for unicode strings.

update(*_=None*)

update_codes(_=*None*)
Read settings from the YAML files and populate self.database with them.

update_computers(_=*None*)

update_settings(_=*None*)
Update code settings.

class aiidalab_widgets_base.**CodeDropdown**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Code selection widget. Attributes:

selected_code(Unicode or Code): Trait that points to the selected Code instance. It can be set either to an AiiDA Code instance or to a code label (will automatically be replaced by the corresponding Code instance). It is linked to the 'value' trait of the *self.dropdown* widget.

codes(Dict): Trait that contains a dictionary (label => Code instance) for all codes found in the AiiDA database for the selected plugin. It is linked to the 'options' trait of the *self.dropdown* widget.

allow_hidden_codes(Bool): Trait that defines whether to show hidden codes or not.

allow_disabled_computers(Bool): Trait that defines whether to show codes on disabled computers.

__annotations__ = {}

__init__(*input_plugin*, *description='Select code:'*, *path_to_root='../'*, ***kwargs*)
Dropdown for Codes for one input plugin.

input_plugin (str): Input plugin of codes to show.

description (str): Description to display before the dropdown.

__module__ = 'aiidalab_widgets_base.codes'

static _full_code_label(*code*)

_get_codes()
Query the list of available codes.

_trait_default_generators = {}

_validate_selected_code

allow_disabled_computers
A boolean (True, False) trait.

allow_hidden_codes
A boolean (True, False) trait.

codes
An instance of a Python dict.

One or more traits can be passed to the constructor to validate the keys and/or values of the dict. If you need more detailed validation, you may use a custom validator method.

Changed in version 5.0: Added key_trait for validating dict keys.

Changed in version 5.0: Deprecated ambiguous `trait`, `traits` args in favor of `value_trait`, `per_key_traits`.

refresh(_=*None*)
Refresh available codes.

The job of this function is to look in AiiDA database, find available codes and put them in the dropdown attribute.

selected_code
A trait type representing a Union type.

class aiidalab_widgets_base.**ComputerDatabaseWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.HBox

Extract the setup of a known computer from the AiiDA code registry.

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_observe_proxy_command

_trait_default_generators = {}

allow_agent
A boolean (True, False) trait.

append_text
A trait for unicode strings.

description
A trait for unicode strings.

hostname
A trait for unicode strings.

label
A trait for unicode strings.

mpiprocs_per_machine
An int trait.

mpirun_command
A trait for unicode strings.

num_cores_per_mpiproc
An int trait.

port
An int trait.

prepend_text
A trait for unicode strings.

proxy_command
A trait for unicode strings.

proxy_hostname
A trait for unicode strings.

proxy_username
A trait for unicode strings.

queue_name
A trait for unicode strings.

safe_interval
A float trait.

scheduler
A trait for unicode strings.

shebang
A trait for unicode strings.

transport
A trait for unicode strings.

update(*_=None*)

update_computers(*_=None*)

update_settings(*_=None*)
Read settings from the YAML files and populate self.database with them.

use_login_shell
A boolean (True, False) trait.

work_dir
A trait for unicode strings.

class aiidalab_widgets_base.**ComputerDropdown**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select a configured computer.

Attributes:

selected_computer(Unicode or Computer): Trait that points to the selected Computer instance.
It can be set either to an AiiDA Computer instance or to a computer label (will automatically be replaced by the corresponding Computer instance). It is linked to the 'value' trait of *self._dropdown* widget.

computers(Dict): Trait that contains a dictionary (label => Computer instance) for all computers found in the AiiDA database. It is linked to the 'options' trait of *self._dropdown* widget.

allow_select_disabled(Bool): Trait that defines whether to show disabled computers.

__annotations__ = {}

__init__(*description='Select computer:'*, *path_to_root='../'*, ***kwargs*)
Dropdown for configured AiiDA Computers.

description (str): Text to display before dropdown.

path_to_root (str): Path to the app's root folder.

__module__ = 'aiidalab_widgets_base.computers'

_get_computers()
Get the list of available computers.

_trait_default_generators = {}

_validate_selected_computer

allow_select_disabled
A boolean (True, False) trait.

computers
An instance of a Python dict.

One or more traits can be passed to the constructor to validate the keys and/or values of the dict. If you need more detailed validation, you may use a custom validator method.

Changed in version 5.0: Added key_trait for validating dict keys.

Changed in version 5.0: Deprecated ambiguous `trait`, `traits` args in favor of `value_trait`, `per_key_traits`.

refresh(*_=None*)
Refresh the list of configured computers.

selected_computer
A trait type representing a Union type.

class aiidalab_widgets_base.**ExportButtonWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_button.Button

Export Node button.

__annotations__ = {}

__init__(*process*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.export'

_trait_default_generators = {}

export_aiida_subgraph(*change=None*)
Perform export when the button is pressed

class aiidalab_widgets_base.**MultiStructureUploadWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to deal with archives (tar or zip) containing multiple structures.

__annotations__ = {}

__init__(*text='Upload Zip or Tar archive'*, *node_class=None*, ***kwargs*)
Upload multiple structures and store them in AiiDA database.

Parameters

- **text** (*str*) – Text to display before upload button
- **node_class** – AiiDA node class for storing the structure. Possible values: 'StructureData', 'CifData' or None (let the user decide). Note: If your workflows require a specific node class, better fix it here.

__module__ = 'aiidalab_widgets_base.structures_multi'

_on_click_store_all(*change*)
Store all the uploaded structures.

_on_click_store_selected(*change*)

_on_file_upload(*change*)
Process the archive once it is uplodaded.

_trait_default_generators = {}

change_structure()

get_ase(*filepath*)
Get an ase object containing the structure.

static get_description(*structure_ase*, *filepath*)
Get the structure description automatically.

property node_class

refresh_view()
Refresh the structure view.

select_structure(*filepath*)
Perform structure selection.

store_structure(*filepath*, *description=None*)
Store the structure in the AiiDA database.

class aiidalab_widgets_base.**NodesTreeWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_output.Output

A tree widget for the structured representation of a nodes graph.

NODE_TYPE = {<class 'aiida.orm.nodes.process.workflow.workchain.WorkChainNode'>: <cla

PROCESS_STATE_STYLE = {<ProcessState.EXCEPTED: 'excepted'>: 'danger', <ProcessState.F

PROCESS_STATE_STYLE_DEFAULT = 'default'

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.nodes'

classmethod _build_tree(*root*)
Recursively build a tree nodes graph for a given tree node.

_convert_to_tree_nodes(*old_nodes*, *new_nodes*)
Convert nodes into tree nodes while re-using already converted nodes.

classmethod _find_called(*root*)

classmethod _find_children(*root*)
Find all children of the provided AiiDA node.

classmethod _find_outputs(*root*)

_observe_nodes

_observe_tree_selected_nodes(*change*)

_refresh_output()

classmethod _to_tree_node(*node*, *name=None*, ***kwargs*)
Convert an AiiDA node to a tree node.

_trait_default_generators = {}

_update_tree_node(*tree_node*)

classmethod _walk_tree(*root*)
Breadth-first search of the node tree.

find_node(*pk*)

nodes
An instance of a Python tuple.

selected_nodes
An instance of a Python tuple.

update(*_=None*)
Refresh nodes based on the latest state of the root process and its children.

class aiidalab_widgets_base.**OptimadeQueryWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

AiiDAlab-specific OPTIMADE Query widget

Useful as a widget to integrate with the *aiidalab_widgets_base.structures.StructureManagerWidget*, embedded into applications.

NOTE: *embedded* for *OptimadeQueryFilterWidget* was introduced in *optimade-client* version 2020.11.5.

Parameters

- **embedded** (*bool*) – Whether or not to show extra database and provider information. When set to *True*, the extra information will be hidden, this is useful in situations where the widget is used in a Tab or similar, e.g., for the *aiidalab_widgets_base.structures.StructureManagerWidget*.
- **title** (*str*) – Title used for Tab header if employed in *aiidalab_widgets_base.structures.StructureManagerWidget*.

__annotations__ = {}

__init__(*embedded: bool = True*, *title: Optional[str] = None*, ***kwargs*) → None
Public constructor

__module__ = 'aiidalab_widgets_base.databases'

_trait_default_generators = {}

_update_structure(*change: dict*) → None
New structure chosen

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**ProcessCallStackWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_string.HTML

Widget that shows process call stack.

__annotations__ = {}

__init__(*title='Process Call Stack'*, *path_to_root='../'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

calc_info(*node*)
Return a string with the summary of the state of a CalculationNode.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the call stack that is shown.

class aiidalab_widgets_base.**ProcessFollowerWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A Widget that follows a process until finished.

__annotations__ = {}

__init__(*process=None*, *followers=None*, *update_interval=0.1*, *path_to_root='../'*, ***kwargs*)
Initiate all the followers.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

follow(*detach=False*)
Initiate following the process with or without blocking.

on_completed(*function*)
Run functions after a process has been completed.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()

class aiidalab_widgets_base.**ProcessInputsWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select and show process inputs.

__annotations__ = {}

__init__(*process=None*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

show_selected_input(*change=None*)
Function that displays process inputs selected in the *inputs* Dropdown widget.

class aiidalab_widgets_base.**ProcessListWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

List of AiiDA processes.

past_days (int): Sumulations that were submitted in the last *past_days*.

incoming_node (int, str, Node): Trait that takes node id or uuid and returns the node that must be among the input nodes of the process of interest.

outgoing_node (int, str, Node): Trait that takes node id or uuid and returns the node that must be among the output nodes of the process of interest.

process_states (list): List of allowed process states.

process_label (str): Show process states of type *process_label*.

description_contains (str): string that should be present in the description of a process node.

__annotations__ = {}

__init__(*path_to_root='../'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_default_process_label

_follow(*update_interval*)

_trait_default_generators = {'process_label': <traitlets.traitlets.DefaultHandler obj

_validate_incoming_node

_validate_outgoing_node

_validate_process_label

description_contains
A trait for unicode strings.

incoming_node
A trait type representing a Union type.

outgoing_node
A trait type representing a Union type.

past_days
An int trait.

process_label
A trait for unicode strings.

process_states
An instance of a Python list.

start_autoupdate(*update_interval=10*)

update(*_=None*)
Perform the query.

class aiidalab_widgets_base.**ProcessMonitor**(***kwargs*)
Bases: traitlets.traitlets.HasTraits

Monitor a process and execute callback functions at specified intervals.

__annotations__ = {}

__init__(*callbacks=None*, *on_sealed=None*, *timeout=None*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

__module__ = 'aiidalab_widgets_base.process'

_monitor_process(*process_id*)

_observe_process

_trait_default_generators = {}

join()

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**ProcessNodesTreeWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A tree widget for the structured representation of a process graph.

__annotations__ = {}

__init__(*title='Process Tree'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_observe_process

_observe_tree_selected_nodes(*change*)

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

selected_nodes
An instance of a Python tuple.

update(*_=None*)

class aiidalab_widgets_base.**ProcessOutputsWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Widget to select and show process outputs.

__annotations__ = {}

__init__(*process=None*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

show_selected_output(*change=None*)
Function that displays process output selected in the *outputs* Dropdown widget.

class aiidalab_widgets_base.**ProcessReportWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_string.HTML

Widget that shows process report.

__annotations__ = {}

__init__(*title='Process Report'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update report that is shown.

class aiidalab_widgets_base.**ProgressBarWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

A bar showing the proggress of a process.

__annotations__ = {}

__init__(*title='Progress Bar'*, ***kwargs*)
Initialize ProgressBarWidget.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

property current_state

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the bar.

class aiidalab_widgets_base.**RunningCalcJobOutputWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Show an output of selected running child calculation.

__annotations__ = {}

__init__(*title='Running Job Output'*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

update()
Update the displayed output.

class aiidalab_widgets_base.**SmilesWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Conver SMILES into 3D structure.

SPINNER = ''

__annotations__ = {}

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_default_structure

_on_button_pressed(*change*)
Convert SMILES to ase structure when button is pressed.

_pybel_opt(*smile*, *steps*)
Optimize a molecule using force field and pybel (needed for complex SMILES).

_rdkit_opt(*smile*, *steps*)
Optimize a molecule using force field and rdkit (needed for complex SMILES).

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

make_ase(*species*, *positions*)
Create ase Atoms object.

mol_from_smiles(*smile*, *steps=10000*)
Convert SMILES to ase structure try rdkit then pybel

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**SshComputerSetup**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Setup password-free access to a computer.

__annotations__ = {}

__init__(***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.computers'

property __password
Returning the password and immediately destroying it

property __private_key
unwrap private key file and setting filename and file content

property __proxy_password
Returning the password and immediately destroying it

static _add_private_key(*private_key_fname*, *private_key_content*)
param private_key_fname: string param private_key_content: bytes

_configure_proxy(*password*, *proxy_password*)
Configure proxy server.

_make_host_known(*hostname*, *proxycmd=None*)
Add host information into known_hosts file.

_observe_proxy_hostname

_observe_proxy_username

_on_setup_ssh(*mode*, *change*)
ATTENTION: modifying the order of operations in this function can lead to unexpected problems

static _send_pubkey(*hostname*, *username*, *password*, *proxycmd=''*)
Send a publick key to a remote host.

static _ssh_keygen()
Generate ssh key pair.

_trait_default_generators = {}

_validate_port

_write_ssh_config(*proxycmd=''*, *private_key_abs_fname=None*)
Put host information into the config file.

can_login(*silent=False*)
Check if it is possible to login into the remote host.

hostname
A trait for unicode strings.

is_host_known(*hostname=None*)
Check if the host is known already.

is_in_config()
Check if the config file contains host information.

on_setup_ssh(*change*)
Setup ssh, password and private key are supported

on_use_diff_proxy_username_change(*change*)
If using different username for proxy check-box is clicked.

on_use_proxy_change(*change*)
If proxy check-box is clicked.

on_use_verification_mode_change(*change*)
which verification mode is chosen.

port
A trait type representing a Union type.

proxy_hostname
A trait for unicode strings.

proxy_username
A trait for unicode strings.

setup_counter
An int trait.

use_proxy
A boolean (True, False) trait.

username
A trait for unicode strings.

class aiidalab_widgets_base.**StructureBrowserWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to query for structures stored in the AiiDA database.

__annotations__ = {}

__init__(*title=''*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_on_select_structure(*_=None*)

_trait_default_generators = {}

preprocess()
Search structures in AiiDA database and add formula extra to them.

search(*_=None*)
Launch the search of structures in AiiDA database.

structure
A trait type representing a Union type.

class aiidalab_widgets_base.**StructureExamplesWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class to provide example structures for selection.

__annotations__ = {}

__init__(*examples*, *title=''*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_default_structure

_on_select_structure(*change*)
When structure is selected.

_trait_default_generators = {'structure': <traitlets.traitlets.DefaultHandler object>

static get_example_structures(*examples*)
Get the list of example structures.

structure
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**StructureManagerWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Upload a structure and store it in AiiDA database.

Attributes: structure(Atoms): trait that contains the selected structure. 'None' if no structure is selected. structure_node(StructureData, CifData): trait that contains AiiDA structure object node_class(str): trait that contains structure_node type (as string).

SUPPORTED_DATA_FORMATS = {'CifData': 'cif', 'StructureData': 'structure'}

__annotations__ = {}

__init__(*importers*, *viewer=None*, *editors=None*, *storable=True*, *node_class=None*, ***kwargs*)

Arguments:

importers(list): list of tuples each containing the displayed name of importer and the importer object. Each object should contain 'structure' trait pointing to the imported structure. The trait will be linked to 'structure' trait of this class.

storable(bool): Whether to provide Store button (together with Store format)

node_class(str): AiiDA node class for storing the structure. Possible values: 'StructureData', 'CifData' or None (let the user decide). Note: If your workflows require a specific node class, better fix it here.

__module__ = 'aiidalab_widgets_base.structures'

_change_structure_node

_convert_to_structure_node(*structure*)
Convert structure of any type to the StructureNode object.

_default_node_class

_observe_input_structure

_observe_structure_node

_structure_changed

_structure_editors(*editors*)
Preparing structure editors.

_structure_importers(*importers*)
Preparing structure importers.

_sync_structure_node()
Synchronize the structure_node trait using the currently provided info.

_trait_default_generators = {}

input_structure
A trait type representing a Union type.

node_class
A trait for unicode strings.

store_structure(*_=None*)
Stores the structure in AiiDA database.

structure
A trait type representing a Union type.

structure_node
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

undo(*_*)
Undo modifications.

class aiidalab_widgets_base.**StructureUploadWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Class that allows to upload structures from user's computer.

__annotations__ = {}

__init__(*title=''*, *description='Upload Structure'*)
Public constructor

__module__ = 'aiidalab_widgets_base.structures'

_on_file_upload(*change=None*)
When file upload button is pressed.

_trait_default_generators = {}

_validate_and_fix_ase_cell(*ase_structure*, *vacuum_ang=10.0*)
Checks if the ase Atoms object has a cell set, otherwise sets it to bounding box plus specified "vacuum" space

structure
A trait type representing a Union type.

class aiidalab_widgets_base.**SubmitButtonWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

Submit button class that creates submit button jupyter widget.

__annotations__ = {}

__init__(*process_class*, *inputs_generator=None*, *input_dictionary_function=None*, *description='Submit'*, *disable_after_submit=True*, *append_output=False*, ***kwargs*)
Submit Button widget.

process_class (Process): Process class to submit.

inputs_generator (func): Function that returns inputs dictionary or inputs builder.

input_dictionary_function (DEPRECATED): Function that generates input parameters dictionary.

description (str): Description written on the submission button.

disable_after_submit (bool): Whether to disable the button after the process was submitted.

append_output (bool): Whether to clear widget output for each subsequent submission.

__module__ = 'aiidalab_widgets_base.process'

_trait_default_generators = {}

on_btn_submit_press(*_=None*)
When submit button is pressed.

on_click(*function*)

on_submitted(*function*)
Run functions after a process has been submitted successfully.

process
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class aiidalab_widgets_base.**WizardAppWidget**(***kwargs*)
Bases: ipywidgets.widgets.widget_box.VBox

ICONS = {<State.INIT: 0>: '', <State.READY: 2>: '', <State.CONFIGURED: 1>: '', <

ICON_SEPARATOR = '\u2000'

__annotations__ = {}

__init__(*steps*, ***kwargs*)
Public constructor

__module__ = 'aiidalab_widgets_base.wizard'

_consider_auto_advance(*_=None*)
Determine whether the app should automatically advance to the next step.

This is performed whenever the current step is within the SUCCESS state and has the auto_advance attribute set to True.

_observe_selected_index

_on_click_back_button(*_*)

_on_click_next_button(*_*)

_on_click_reset_button(*_*)

_trait_default_generators = {}

_update_buttons()

_update_step_state(*_*)

_update_titles()

can_reset()

classmethod icons()
Return the icon set and return animated icons based on the current time stamp.

reset(*step=0*)
Reset the app up to the given step.

For example, with step=0 (the default), the whole app is reset. With step=1, all but the first step are reset.

selected_index
An int trait.

class aiidalab_widgets_base.**WizardAppWidgetStep**(***kwargs*)
Bases: traitlets.traitlets.HasTraits

One step of a WizardAppWidget.

class State(*value*)
Bases: enum.Enum

Each step is always in one specific state.

The state is used to determine:

1) how the step is visually presented to the user, and

2) whether the next step is accessible (i.e. reached the SUCCESS state).

App developers are encouraged to use the step states to couple application logic and interface. In general, all widget changes should trigger a re-evaluation of the step state, and states also determine whether certain widgets are enabled or disabled.

A step can be in one of the following states:

INIT: The initial state, usually all widgets disabled. READY: The step (widget) is ready for user input (some or all widgets enabled). CONFIGURED: The step is in a consistent configuration awaiting confirmation. ACTIVE: The step is carrying out a runtime operation. SUCCESS: A

configuration has been confirmed / a runtime operation successfully finished. FAIL: A runtime operation has failed in an unrecoverable way.

Not all steps must implement all states, for example:

- the first step does not need an INIT state
- a step without runtime process should not have an ACTIVE or FAIL state
- a "review & confirm" step does not require a READY state.
- a step without configuration options (e.g. pure "review & confirm" step)

Important: The next step is only accessible if the current step is within the SUCCESS state!

ACTIVE = 3

CONFIGURED = 1

FAIL = -1

INIT = 0

READY = 2

SUCCESS = 4

__annotations__ = {}

__module__ = 'aiidalab_widgets_base.wizard'

__annotations__ = {}

__module__ = 'aiidalab_widgets_base.wizard'

_trait_default_generators = {}

auto_advance
A boolean (True, False) trait.

can_reset()

state
Use a Enum class as model for the data type description. Note that if no default-value is provided, the first enum-value is used as default-value.

```
# -- SINCE: Python 3.4 (or install backport: pip install enum34)
import enum
from traitlets import HasTraits, UseEnum

class Color(enum.Enum):
    red = 1         # -- IMPLICIT: default_value
    blue = 2
    green = 3

class MyEntity(HasTraits):
    color = UseEnum(Color, default_value=Color.blue)

entity = MyEntity(color=Color.red)
entity.color = Color.green    # USE: Enum-value (preferred)
entity.color = "green"        # USE: name (as string)
entity.color = "Color.green"  # USE: scoped-name (as string)
entity.color = 3              # USE: number (as int)
assert entity.color is Color.green
```

aiidalab_widgets_base.**register_viewer_widget**(*key*)
Register widget as a viewer for the given key.

aiidalab_widgets_base.**viewer**(*obj*, *downloadable=True*, ***kwargs*)
Display AiiDA data types in Jupyter notebooks.

Parameters **downloadable** (*bool*) – If True, add link/button to download the content of displayed AiiDA object.

Returns the object itself if the viewer wasn't found.

aiidalab-widgets-base package is released under the MIT license.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

5.1 How to cite

Users of AiiDAlab are kindly asked to cite the following publication in their own work:

A. V. Yakutovich et al., Comp. Mat. Sci. 188, 110165 (2021). DOI:10.1016/j.commatsci.2020.110165

5.2 Acknowledgements

This work is supported by the MARVEL National Centre for Competency in Research funded by the Swiss National Science Foundation, as well as by the MaX European Centre of Excellence funded by the Horizon 2020 EINFRA-5 program, Grant No. 676598.

PYTHON MODULE INDEX

a

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W