***rospy overview (/rospy/Overview)**: Initialization and Shutdown (/rospy/Overview/Initialization%20and%20Shutdown) | Messages (/rospy/Overview/Messages) | Publishers and Subscribers (/rospy/Overview/Publishers%20and%20Subscribers) | Services (/rospy/Overview/Services) | Parameter Server | Logging (/rospy/Overview/Logging) | Names and Node Information (/rospy/Overview/Names%20and%20Node%20Information) | Time (/rospy/Overview/Time) | Exceptions (/rospy/Overview/Exceptions) | tf/Overview (/tf/Overview) | tf/Tutorials (/tf/Tutorials) | Python Style Guide (/PyStyleGuide)*

Sommaire

The ROS Parameter Server (/Parameter%20Server) can store strings, integers, floats, booleans, lists, dictionaries, iso8601 dates, and base64-encoded data. Dictionaries must have string keys.

rospy's API is a thin wrapper around Python's builtin xmlrpclib (https://docs.python.org/3/library/xmlrpc.html) library, so you should consult that documentation on how to properly encode values. In most cases (booleans, ints, floats, strings, arrays, dictionaries), you can use values as-is.

NOTE: parameter server methods are not threadsafe, so if you are using them from multiple threads, you must lock appropriate.

1. Getting parameters

`rospy.get_param(param_name)`

> Fetch value from the Parameter Server (/Parameter%20Server). You can optionally pass in a default value to use if the parameter is not set. Names are resolved relative to the node's namespace. If you use `get_param()` to fetch a namespace, a dictionary is returned with the keys equal to the parameter values in that namespace. `KeyError` is raised if the parameter is not set.

Afficher/masquer les numéros de lignes

```
global_name = rospy.get_param("/global_name")
relative_name = rospy.get_param("relative_name")
private_param = rospy.get_param('~private_name')
default_param = rospy.get_param('default_param', 'default_value'
)

# fetch a group (dictionary) of parameters
gains = rospy.get_param('gains')
p, i, d = gains['p'], gains['i'], gains['d']
```

2. Setting parameters

As described earlier, you can set parameters to store strings, integers, floats, booleans, lists, and dictionaries. You can also use iso8601 dates and base64-encoded data, though those types are discouraged as they are not commonly used in other ROS client libraries. Dictionaries must have string keys as they are considered to be *namespaces* (see example below).

`rospy.set_param(param_name, param_value)`

Set parameter on the Parameter Server. Names are resolved relative to the node's namespace.

Afficher/masquer les numéros de lignes

```
# Using rospy and raw python objects
rospy.set_param('a_string', 'baz')
rospy.set_param('~private_int', 2)
rospy.set_param('list_of_floats', [1., 2., 3., 4.])
rospy.set_param('bool_True', True)
rospy.set_param('gains', {'p': 1, 'i': 2, 'd': 3})

# Using rosparam and yaml strings
rosparam.set_param('a_string', 'baz')
rosparam.set_param('~private_int', '2')
rosparam.set_param('list_of_floats', "[1., 2., 3., 4.]")
rosparam.set_param('bool_True', "true")
rosparam.set_param('gains', "{'p': 1, 'i': 2, 'd': 3}")

rospy.get_param('gains/p') #should return 1
```

3. Parameter existence

Testing for parameter existence is useful if you wish to save network bandwidth transferring the parameter value or if you don't wish to use `try/except` blocks with `get_param` and `delete_param`.

`rospy.has_param(param_name)`

Return `True` if parameter is set, `False` otherwise.

Afficher/masquer les numéros de lignes

```
1 if rospy.has_param('to_delete'):
2     rospy.delete_param('to_delete')
```

4. Deleting parameters

`rospy.delete_param(param_name)`

Delete the parameter from the Parameter Server (/Parameter%20Server). The parameter must be set (`KeyError` is raised if not set). Names are resolved relative to the node's namespace.

Afficher/masquer les numéros de lignes

```
1 try:
2     rospy.delete_param('to_delete')
3 except KeyError:
4     print("value not set")
```

5. Retrieve list of parameter names

New in ROS indigo

`rospy.get_param_names()`

You can get list of existing parameter names as list of strings with the `get_param_names()` interface.

Afficher/masquer les numéros de lignes

```
1 try:
2     rospy.get_param_names()
3 except ROSException:
4     print("could not get param name")
```

6. Searching for parameter keys

There are times where you want to get a parameter from the *closest* namespace. For example, if you have a "`robot_name`" parameter, you just want to search upwards from your private namespace until you find a matching parameter. Similarly, if you have a group of camera nodes, you may wish to

set some parameters commonly in a shared namespace but override others by setting them in private (~name) namespace.

NOTE: in order to use `search_param` effectively, you should use it with *relative* names instead of `/global` and `~private` names.

`rospy.search_param(param_name)`

> Find closest parameter name, starting in the private namespace and searching upwards to the global namespace. Returns `None` if no match found.

Afficher/masquer les numéros de lignes

```
1 param_name = rospy.search_param('global_example')
2 v = rospy.get_param(param_name)
```

If this code appears in the node `/foo/bar`, `rospy.search_param` will try to find the parameters

1. `/foo/bar/global_example`
2. `/foo/global_example`
3. `/global_example`

in this order.

7. Getting parameter names

`rospy.get_param_names()`

> Return a list of all the parameter names on the Parameter Server (/Parameter%20Server).

Wiki: rospy/Overview/Parameter Server (dernière édition le 2021-01-11 06:56:39 par AvneeshMishra (mailto:123avneesh@gmail.com))