reapy Documentation

Release 0.10.0

Roméo Després

Dec 29, 2020

Contents

Contents

`reapy` is a nice pythonic wrapper around the quite unpythonic ReaScript Python API for REAPER.

CHAPTER 1

Installation

See the Installation guide for installation instructions.

CHAPTER 2

Usage

reapy has two main features: it allows nice pythonic code, and interactions with REAPER from the outside. Instead of creating a new ReaScript containing:

```
>>> from reaper_python import *
>>> RPR_ShowConsoleMsg("Hello world!")
```

you can open your usual Python shell and type:

```
>>> import reapy
>>> reapy.print("Hello world!")
```

2.1 ReaScript API

All ReaScript API functions are available in reapy in the sub-module reapy.reascript_api. Note that in ReaScript Python API, all function names start with "RPR_". That unnecessary pseudo-namespace has been removed in reapy. Thus, you shall call reapy.reascript_api.GetCursorPosition in order to trigger reaper_python.RPR_GetCursorPosition. See example below:

```
>>> from reapy import reascript_api as RPR
>>> RPR.GetCursorPosition()
0.0
>>> RPR.SetEditCurPos(1, True, True)
>>> RPR.GetCursorPosition()
1.0
```

2.2 reapy API

The purpose of reapy is to provide a more pythonic API as a substitute for ReaScript API. Below is the reapy way of executing the example above:

```
>>> import reapy
>>> project = reapy.Project() # current project
>>> project.cursor_position
0.0
>>> project.cursor_position = 1
>>> project.cursor_position
1.0
```

The Translation table matches ReaScript functions with their `reapy` counterparts.

2.3 Performance

When used from inside REAPER, `reapy` has almost identical performance than native ReaScript API. Yet when it is used from the outside, the performance is quite worse. More precisely, since external API calls are processed in a `defer` loop inside REAPER, there can only be around 30 to 60 of them per second. In a time-critical context, you should make use of the `reapy.inside_reaper` context manager.

```
>>> import reapy
>>> project = reapy.Project() # Current project
>>>
>>> # Unefficient (and useless) call
>>> bpms = [project.bpm for _ in range(1000)] # Takes at least 30 seconds...
>>>
>>> # Efficient call
>>> with reapy.inside_reaper():
...     bpms = [project.bpm for _ in range(1000)]
...
>>> # Takes only 0.1 second!
```

Although this method should be sufficient in most cases, note that optimality is only reached by making use of `reapy.tools.Program` (see documentation here).

CHAPTER 3

API documentation

Check out the API guide and the Translation table for more information about how to use `reapy`.

CHAPTER 4

Contributing

For now, about a third of ReaScript API has a `reapy` counterpart, the docs are far from great, and many bugs are waiting to be found. Feel free to improve the project by checking the contribution guide !

CHAPTER 5

Author

Roméo Després - RomeoDespres

CHAPTER 6

License

This project is licensed under the MIT License - see the LICENSE.txt file for details.