wirerope

Release 0.4.6

Feb 17, 2022

Contents:

CHAPTER 1

wirerope

1.1 wirerope — Universal method/function wrapper.

1.2 wirerope.wire — end-point instant for each bound method

class wirerope.wire.**Wire**(*rope*, *owner*, *binding*)

The core data object for each function for bound method.

Inherit this class to implement your own Wire classes.

- For normal functions, each function is directly wrapped by **Wire**.
- For any methods or descriptors (including classmethod, staticmethod), each one is wrapped by `wirerope.rope.MethodRopeMixin` and it creates **Wire** object for each bound object.

1.3 wirerope.rope — Wire access dispatcher for descriptor type.

class wirerope.rope.**WireRope**(*wire_class*, *core_class=<class 'wirerope.rope.RopeCore'>*, *wraps=False*, *rope_args=None*)

The end-user wrapper for callables.

Any callable can be wrapped by this class regardless of its concept - free function, method, property or even more weird one. The calling type is decided by each call and redirected to proper RopeMixin.

The rope will detect method or property owner by needs. It also will return or call their associated *wirerope.wire.Wire* object - which are the user defined behavior.

class wirerope.rope.**RopeCore**(*callable*, *rope*)

The base rope object.

To change rope behavior, create a subclass or compatible class and pass it to *core_class* argument of :class wirerope.rope.WireRope‘.

The concepts:

- *wirerope.rope.WireRope* is a wrapper interface for python callable.
- Custom *wirerope.wire.Wire* class provides user-defined behavior. A subclass of this class is working similar to a *decorator function* body.
- A wire object is associated with a bound method.
- Rope is dispatching types.

wirerope.rope.WireRope is the wrapper for callables. By wrapping a function with *WireRope* with a custom subclass of the *wirerope.wire.Wire* class, the wire object will be created by each function or bound method.

Wire is the most important part. The given class will be instantiated and bound to each function or bound method - which fits the concept of *instance cmethod* of human. For example, when *f* is a free function or staticmethod, the wire also will be a single object. When *f* is a method or property, wires will be created for each method owner object *self*. When *f* is a classmethod, wires will be created for each method owner class *cls*. Yes, it will detect the owner and bound to it regardless of the calling type.

Rope is internal dispatcher. It will be helpful when creating a complex object for decorated callable instead of simple callable feature.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

R

W