hoonds Documentation

Release 0.0.11

Pat Daburu

Feb 27, 2020

Contents:

CHAPTER 1

API Documentation

1.1 hoonds

hoonds is an unassuming collection of tools for python.

1.2 hoonds.logging

1.3 hoonds.patterns

python implementations of common design patterns.

1.4 hoonds.patterns.observer

Here we have classes and utilities for implementing the observer pattern.

class hoonds.patterns.observer.**Observable**

Bases: object

Extend this class to implement the observer pattern.

__init__

Initialize self. See help(type(self)) for accurate signature.

send_signal(*signal: enum.Enum*, *args: hoonds.patterns.observer.SignalArgs = None*)

Send a signal to any interested parties.

Parameters

- **signal** (Enum) – the signal
- **args** (Any) – the arguments to the receiver

signals
Get this observable's enumeration of signals. Subclasses should override the property to return their particular enumeration of signals.

Returns the enumeration that defines the signals

Return type Enum

subscribe(*signal: enum.Enum, receiver: typing.Callable[[hoonds.patterns.observer.SignalArgs], NoneType], weak: bool = True*) → hoonds.patterns.observer.SubscriberHandle
Subscribe to a signal.

Parameters

- **signal** (Enum) – this is the signal to which you're subscribing
- **receiver** (Callable[[SignalArguments]]) – this is the function or method that will handle the dispatch
- **weak** (bool) – When True the dispatcher maintains a weak reference to the receiver which will not prevent garbage collection.

Returns a handle that can be used to unsubscribe from the signal

Return type *SubscriberHandle*

Note: If the receiver is a lambda function, or otherwise might need to receive signals after it goes out of scope, you likely want the weak parameter to be True, however you must remember that any time this argument is True you are responsible for making sure the receiver is unsubscribed to prevent memory leaks.

---

unsubscribe_all()
Disconnect all the receivers.

class hoonds.patterns.observer.**SignalArgs**(*data: typing.Dict[str, typing.Any] = None*)
Bases: collections.abc.Mapping

This is a read-only dictionary that holds signal arguments as they're transmitted to receivers.

Note: You can extend this class to provide explicit properties, however you are advised to store the underlying values in the dictionary.

__init__(*data: typing.Dict[str, typing.Any] = None*)

Parameters **data** (Dict[str, Any]) – the arguments

get(*key*) → typing.Any
Get the value of a property by its name. :param key: the property name :return: the value

is_empty_set
Is the argument set empty?

Returns True if the argument set is empty, otherwise False.

Return type bool

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

values() → an object providing a view on D's values

class hoonds.patterns.observer.**SignalsEnum**
Bases: object

This is a flag interface applied to classes decorated with the *signals()* decorator.

__init__
Initialize self. See help(type(self)) for accurate signature.

class hoonds.patterns.observer.**SubscriberHandle**(*signal: enum.Enum, receiver: typing.Callable[[hoonds.patterns.observer.SignalArgs], NoneType], sender: typing.Any*)
Bases: object

This is a handle object that is returned when you subscribe to a signal. It can be used to unsubscribe when you're no longer interested in receiving the dispatches.

__init__(*signal: enum.Enum, receiver: typing.Callable[[hoonds.patterns.observer.SignalArgs], NoneType], sender: typing.Any*)

receiver
Get the receiver.

Return type Callable

sender
Get the sender.

Return type Any

signal
Get the signal.

Return type Enum

unsubscribe()
Unsubscribe from the signal.

hoonds.patterns.observer.**signals**()
Use this decorator to identify the enumeration within your observable class that defines the class' signals.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

Symbols

G

H

I

K

O

R

S

U

V