Speakers Documentation

Release 0.1.0

Gabriel Falcão

Apr 21, 2018

Contents

Contents:

CHAPTER 1

API Reference

CHAPTER 2

Tutorial

Installing

```
pip install speakers
```

Basic Usage

```
import sure
from mock import Mock
from speakers import Speaker

after = Speaker('after', ['getting_sql_results'])
```

Full example

```
on = Speaker('on', ['ready', 'loading'])

@on.ready
def show_ready(event, dom):
    print "The DOM is ready!"
    print "The page title is", dom.cssselect("title")

@on.loading
def show_loading(event):
    print "Loading DOM, please wait"

on.loading.shout()


fake_dom = Mock()
fake_dom.cssselect.return_value = "A cool title!"

on.loading.shout()
on.ready.shout(fake_dom)

fake_dom.cssselect.assert_called_once_with("title")
```

Multiple handlers

You can have as many handlers as you want, they will be called serially one after the other

```
@after.getting_sql_results
def print_results(event, results):
    print "Doing query results"
    print
    for r in results:
        print "Query result", r
```

Firing events

In speakers it's called `shout()`

```
results = ["res1", "res2", "res3"]
after.getting_sql_results.shout(results)
```

Exception handling

Speakers allows you to declare a single exception handler function per speaker instance.

You can decorate a function that must take 4 arguments:

- `event`: the Speaker instance
- `exc`: the exception instance, you could use the traceback module to print the full exception trace if you want.
- `args`: a tuple containing the arguments passed to the callback which raised the current exception
- `kwargs`: a dictionary containing the keyword arguments passed to the callback which raised the current exception.

Example:

```
on = Speaker('on', ['ready', 'loading'])

@on.exception_handler
def print_exception(event, exc, args, kwargs):
    exc.should.be.a(TypeError)

    print exc

@on.loading
def show_loading(event):
    # this should raise a TypeError
    range(10) + {'foo': 'bar'}

on.loading.shout()
```

Removing event handlers

```
when = Speaker('when', ['ready'])

@when.ready
def never_called(event):
    raise IOError("BOOM")

@when.ready
def called_once(event):
    assert len(event.hooks["ready"]) == 1


when.ready.unplug(never_called)

# calls the function `called_once`
when.ready.shout()


when = Speaker('when', ['ready'])

@when.ready
def never_called(event):
    raise IOError("BOOM")

@when.ready
def called_once(event):
    assert len(event.hooks["ready"]) == 1

when.release('ready')

# nothing will happen
when.ready.shout()
```

Removing all handlers from a speaker

```
when = Speaker('when', ['ready', 'loading'])

@when.ready
def dont_call_me(event):
    raise RuntimeError("You got served")

@when.loading
def do_something(event):
    raise RuntimeError("You got served")

when.release()

# nothing will happen
when.ready.shout()
```

Removing all handlers

```
when = Speaker('when', ['ready', 'loading'])
after = Speaker('after', ['ready', 'loading'])

@when.ready
def dont_call_me(event):
    raise RuntimeError("You got served")

@when.loading
def do_something(event):
    raise RuntimeError("You got served")

@after.ready
def after_all(event):
    raise RuntimeError("You got served")

@after.loading
def loading_after(event):
    raise RuntimeError("You got served")

Speaker.release_all()

when.ready.shout()
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

S

S