Vireo Documentation

Release 1.0a1

Juti Noppornpitak

Jun 21, 2017

Contents

A library and framework for event-driven application development

Warning: This library is currently under active development. Until version 1.0, the API signatures are subject to change. While the code should be stable (not easily throwing exceptions) even before reaching version 1.0, version pinning is highly recommended.

CHAPTER 1

API Reference

Core and Observer

class vireo.observer.**Observer**(*driver*)
Event Observer

id
Observer Identifier

join(*running_mode=2*)
Wait for all handlers to stop.

There are two mode: synchronous (vireo.observer.SYNC_START) and asynchronous (vireo.observer.ASYNC_START) joins.

```
app.join(ASYNC_START)
```

on(*event_name*, *callback*, *resumable=False*, *simple_handling=True*)
Listen to an event with a callback function.

Parameters

- **event_name** (*str*) – the name of the event
- **callback** (*callable*) – the callback callable
- **resumable** (*bool*) – the flag to indicate whether the event consumption can be resumed (as the data stream will never be deleted).
- **simple_handling** (*bool*) – the flag to instruct the code to return the content of the message, instead of returning the whole vireo.model.Message object.

The callback is a callable object, e.g., function, class method, lambda object, which takes only one parameter which is a JSON-decoded object.

For example,

```
def on_foo(self, message):
    print('on_foo:', message)

app.on('foo', on_foo)
app.on('foo.lambda', lambda x: print('foo_lambda:', x))
```

Here is an example for `error_handler`.

```
def error_handler(consumer, exception):
    ...
```

on_broadcast(*event_name*, *callback*, *simple_handling=True*)
Listen to an distributed event with a callback function.

Parameters

- **event_name** (*str*) – the name of the event
- **callback** (*callable*) – the callback callable
- **simple_handling** (*bool*) – the flag to instruct the code to return the content of the message, instead of returning the whole `vireo.model.Message` object.

The callback is a callable object, e.g., function, class method, lambda object, which takes only one parameter which is a JSON-decoded object.

For example,

```
def on_foo(self, message):
    print('on_foo:', message)

app.on('foo', on_foo)
app.on('foo.lambda', lambda x: print('foo_lambda:', x))
```

Here is an example for `error_handler`.

```
def error_handler(consumer, exception):
    ...
```

stop()
Send the signal to all handlers to stop observation.

> **Warning:** This method does not block the caller thread while waiting all handlers to stop.

```
app.stop()
```

exception vireo.observer.**UnknownRunningModeError**
Error for unknown running mode

Exceptions

exception vireo.exception.**NoConnectionError**
No connection error

exception vireo.exception.**ObservationError**
Observation error

RabbitMQ Driver

class vireo.drivers.rabbitmq.driver.**Driver**(*url*, *consumer_classes=None*, *unlimited_retries=False*, *on_connect=None*, *on_disconnect=None*, *on_error=None*)

Driver for RabbitMQ

Parameters

- **url** (*str*) – the URL to the server
- **consumer_classes** (*list*) – the list of *consumer.Consumer*-based classes
- **unlimited_retries** (*bool*) – the flag to disable limited retry count.
- **on_connect** (*callable*) – a callback function when the message consumption begins.
- **on_disconnect** (*callable*) – a callback function when the message consumption is interrupted due to unexpected disconnection.
- **on_error** (*callable*) – a callback function when the message consumption is interrupted due to exception raised from the main callback function.

Here is an example for `on_connect`.

```
def on_connect(consumer = None):
    ...
```

Here is an example for `on_disconnect`.

```
def on_disconnect(consumer = None):
    ...
```

Here is an example for `on_error`.

```
def on_error(exception, consumer = None):
    ...
```

broadcast(*route*, *message*, *options=None*)

Broadcast a message to a particular route.

Parameters

- **route** (*str*) – the route
- **message** (*str*) – the message
- **options** (*dict*) – additional options for basic_publish

join()

Synchronously join all consumers.

publish(*route*, *message*, *options=None*)

Synchronously publish a message

Parameters

- **route** (*str*) – the route
- **message** (*str*) – the message
- **options** (*dict*) – additional options for basic_publish

setup_async_cleanup()
Prepare to cleanly join all consumers asynchronously.

stop_consuming()
Send the signal to stop consumption.

class vireo.drivers.rabbitmq.consumer.**Consumer**(*url*, *route*, *callback*, *shared_stream*, *resumable*, *distributed*, *queue_options*, *simple_handling*, *unlimited_retries=False*, *on_connect=None*, *on_disconnect=None*, *on_error=None*, *controller_id=None*)

Message consumer

This is used to handle messages on one particular route/queue.

Parameters

- **url** (*str*) – the URL to the RabbitMQ server
- **route** (*str*) – the route to observe
- **callback** (*callable*) – the callback function / callable object
- **shared_stream** (*list*) – the internal message queue for thread synchronization
- **resumable** (*bool*) – the flag to indicate whether the consumption is resumable
- **resumable** – the flag to indicate whether the messages are distributed evenly across all consumers on the same route
- **queue_options** (*dict*) – additional queue options
- **unlimited_retries** (*bool*) – the flag to disable limited retry count.
- **on_connect** (*callable*) – a callback function when the message consumption begins.
- **on_disconnect** (*callable*) – a callback function when the message consumption is interrupted due to unexpected disconnection.
- **on_error** (*callable*) – a callback function when the message consumption is interrupted due to exception raised from the main callback function.

Here is an example for `on_connect`.

```
def on_connect(consumer = None):
    ...
```

Here is an example for `on_disconnect`.

```
def on_disconnect(consumer = None):
    ...
```

Here is an example for `on_error`.

```
def on_error(exception, consumer = None):
    ...
```

static can_handle_route(*routing_key*)
Check if the consumer can handle the given routing key.

Note: the default implementation will handle all routes.

Parameters **routing_key** (*str*) – the routing key

stop()
Stop consumption

class vireo.drivers.rabbitmq.exception.**NoConnectionError**
No connection error

class vireo.drivers.rabbitmq.exception.**SubscriptionNotAllowedError**
Subscription not allowed

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

v

Index