EasyRabbit Documentation

Release 0.0.2

scnerd

Feb 27, 2018

Contents:

Actually easy RabbitMQ utilities for common tasks. Hides the complexities of complete control packages like `pika` by wrapping common use cases in minimalistic wrappers.

CHAPTER 1

Installation

You can install directly from Pypi:

```
pip install easyrabbit
```

or from git using pip:

```
pip install git+https://github.com/scnerd/easyrabbit
```

CHAPTER 2

Usage

2.1 Simple routing

2.1.1 Reader

`RoutingReader` provides a fully asynchronous way to read from an exchange via a queue bound with a routing key. It launches a subprocess to free the calling program from being responsible for its computational overhead; within its own process, it uses `pika`'s asynchronous connection, enabling the highest possible performance client. The API exposed mimics a simple queue, if, after all, what you really want is to use a RabbitMQ queue as if it were a Python queue.

```
with RoutingReader(url, exchange, queue_name, routing_key) as reader:
    for msg in reader:
        print("Received the following message: {}".format(msg))
```

If the reader is needed persistently, you can also launch and terminate it yourself:

```
reader = RoutingReader(url, exchange, queue_name, routing_key)
reader.start()
# Do things with the reader
reader.close()
```

Note that even though the reader is asynchronous, you don't need to wait for it to be ready before using it. All calls hang on the process pipe that sends data from the client process to your parent code. If you want to make sure that the connection is fully established before using it, however, you can use `wait_till_ready`:

```
reader = RoutingReader(url, exchange, queue_name, routing_key)
reader.start()
try:
   reader.wait_till_ready(timeout=5)
except TimeoutError:
   raise RuntimeError("RabbitMQ reader took more than 5 seconds to launch")
```

While `reader.get` is blocking, a non-blocking equivalent is `reader.get_nowait`, or your code can explicitly check that a value is available first using `not reader.empty()`. Iterating over `reader` just repeatedly calls `get`, and hence is blocking and will only end when the reader is closed. Use `reader.getall_nowait()` to obtain all values currently in the queue.

All result objects are byte arrays.

2.1.2 Writing

The analogous utility for writing is `RoutingWriter`, which provides a nearly identical interface for writing messages to RabbitMQ

```
with RoutingWriter(url, exchange, routing_key) as writer:
    for msg in msgs:
        writer.put(msg)
```

`RoutingWriter` exposes much the same API as `RoutingReader`, except of course exposing `put` instead of `get`.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search