Pikacon Documentation

Release 0.3.2

Jukka Ojaniemi

March 11, 2016

Contents

target https://travis-ci.org/pingviini/pikacon

Pikacon is a helper library which will reduce the amount of boilerplate your software needs when it is using pika for creating connection to broker and declaring exchanges and queues.

Pikacon provides helper class (ClassName) which can be imported to your program. Class takes a path to ini-style config file as a parameter and creates connection, exchanges, queues and bindings automatically from there. All you need to provide is a proper config.

Contents:

CHAPTER 1

Configuration

Pikacon uses Pythons ConfigParser to get config for connection, exchanges and queues.

1.1 Broker

Broker is configured as in above example. Section name is broker and options are regular pika parameters for broker. If you want to configure ssl_options create new section for those and refer that section name in broker options. Eg.:

```
[broker]
...
ssl = True
ssl_options = my_ssl_options

[my_ssl_options]
...
```

1.2 Exchange

Section name for exchange consists of two parts divided by ':'. First part is 'exchange' and second part is the name of the exchange. Eg.:

```
[exchange:myexchange]
...
```

The actual options below exchange section are normal key = value parameters which are used in pika.

1.3 Binding

The name of the binding section consists three parts divided by ':'. First part is always 'binding'. Second part is the name of the queue we're binding. Third part is name of the exchange where we're binding the queue. Eg.:

```
[binding:myqueue:myexchange]
```

The actual options below binding section are normal key = value parameters which are used in pika.

1.4 Processing order

Pikacon's processing order is following:

1. Connection
2. Channel
3. Exchanges
4. Queues
5. Bindings

1.5 Complete configuration example

```
[broker]
host = localhost
port = 5672
username = guest
password = guest
virtual_host = /
heartbeat = 60

[exchange:exchangename]
type = direct
durable = False
auto_delete = True

[queue:testqueue1]
durable = True
exclusive = False

[queue:testqueue2]
durable = False
exclusive = False

[queue:testqueue3]
durable = True
exclusive = False
arguments = queue:testqueue3:arguments

[queue:testqueue4]
durable = True
exclusive = False

[queue:testqueue3:arguments]
x-message-ttl = 1800000
x-dead-letter-exchange = exchangename
x-dead-letter-routing-key = key4

[binding:testqueue1:exchangename]
routing_key = key1

[binding:testqueue2:exchangename]
routing_key = key2
```

```
[binding:testqueue3:exchangename]
routing_key = key3

[binding:testqueue4:exchangename]
routing_key = key4
```

Above example configures connection to broker at localhost. It defines one direct exchange called exchangename and four queues called testqueue1, testqueue2, testqueue3 and testqueue4. Testqueue3 has extra arguments which define dead letter exchange. All queues are bound to our only exchange with routingkeys key1, key2, key3 and key4.

CHAPTER 2

Pikacon usage example

```
from pikacon.pikacon import BrokerConnection

class PikaconExample(object):

    def __init__(self):
        self.broker = BrokerConnection(
            '/path/to/config.cfg',
            callback=self.broker_ready_callback)

    def broker_ready_callback(self):
        """Do something with the broker."""
        self.broker.channel.basic_consume(self.handle_messages,
                                          'queue')
```

Yes. Above is full example of working code which sets up connections, channels, exchanges, queues and bindings assuming they have been configured in /path/to/congfig.cfg. You can start using self.broker as you would use pika.

CHAPTER 3

Source code

Pikacon's source code and issue tracker is at Github. Fork or send a ticket if you have troube with it.