Rhubarb Documentation

Release

Robert Allen

December 26, 2013

Contents

Contents:

CHAPTER 1

Installation

1.1 Composer

The recommended method of installation is via composer

```
composer require zircote/rhubarb:3.1.*
```

Depending on your selection of connectors you will also need to require or compile the appropriate extension or libraries.

Extensions may be installed with pecl i.e.

```
pecl install mongo
```

Libraries can be included utilizing the composer command

```
composer require predis/predis:master-dev
```

1.2 PECL AMQP

Development of the Official PHP AMQP extension may be found at https://github.com/bkw/pecl-amqp-official as well as stubs and tests.

Installation via pecl:

To build the ext-amqp from source:

CHAPTER 2

Rhubarb

Celery Worker Execution From PHP

Use of Rhubarb is outlined as follows.

Send Task and Wait For Result

```
use \Rhubarb\Exception\TimeoutException;

$rhubarb = new \Rhubarb\Rhubarb($options);

try {
   $task = $rhubarb->sendTask('task.add', array(2,2));
   $task->delay();
   $result = $task->get();
} catch (TimeoutException $e) {
   $log->error('task failed to return in default timelimit [10] seconds');
}
```

Fire And Forget Task

```
use \Rhubarb\Exception\TimeoutException;

$rhubarb = new \Rhubarb\Rhubarb($options);

try {
   $task = $rhubarb->sendTask('task.add', array(2,2));

   $result = $task->delay();
} catch (TimeoutException $e) {
   $log->error('task failed to return in default timelimit [10] seconds');
}
```

Getting Task Status

```
use \Rhubarb\Exception\TimeoutException;

$rhubarb = new \Rhubarb\Rhubarb($options);

$task = $rhubarb->sendTask('task.add', array(2,2));
```

```
$result = $task->delay();

while (!$task->successful()) {
   echo $task->state(), PHP_EOL;
   // You should have some time based break; statement here
}

var_dump($task->get());
```

KWARG Support

```
use \Rhubarb\Exception\TimeoutException;

$rhubarb = new \Rhubarb\Rhubarb($options);
try {
   $task = $rhubarb->sendTask('task.add', array('arg1' => 2, 'arg2' => 2));
   $result = $task->delay();
   var_dump($task->get());
} catch (TimeoutException $e) {
   $log->error('task failed to return in default timelimit [10] seconds');
}
```

Method Specific Queue and/or Exchange

```
use \Rhubarb\Exception\TimeoutException;

$rhubarb = new \Rhubarb\Rhubarb($options);
try {
   $task = $rhubarb->sendTask('task.add', array('arg1' => 2, 'arg2' => 2));
   $task->getMessage()
        ->setPropQueue('priority.high')
        ->setPropExchange('queue.other');
   $result = $task->delay();
   var_dump($task->get());
} catch (TimeoutException $e) {
   $log->error('task failed to return in default timelimit [10] seconds');
}
```

Advanced Task Options At runtime it may become necessary to utilize a different queue, exchange or various runtime options. These options may be passed to the __delay__ method when called:

Supported Options are:

- countdown: (int) The task is guaranteed to be executed at some time after the specified date and time, but not necessarily at that exact time.
- expires: (int) The expires argument defines an optional expiry time, either as seconds after task publish.
- priority: (int) A number between 0 and 9, where 0 is the highest priority. (Supported by: redis)
- utc: (bool) Timestamps are UTC.
- eta: (int) The ETA (estimated time of arrival) in seconds; lets you set a specific date and time that is the earliest time at which your task will be executed.
- errbacks: TBD
- queue: (string) Simple routing (name <-> name) is accomplished using the queue option.
- queue_args: (array) Key-Value option pairs for the queue arguments.
- exchange: (string) Name of exchange (or a kombu.entity.Exchange) to send the message to.

Example

```
$rhubarb = new \Rhubarb\Rhubarb($options);

$res = $rhubarb->sendTask('subtract', array(3, 2));
$res->delay(
    array(
        'queue' => 'priority.high',
        'exchange' => 'subtract_queue'
    )
);
$result = $res->get(2);
$this->assertEquals(1, $result);
```

CHAPTER 3

Rhubarb Connectors

Contents:

3.1 AMQP

Rhubarb currently supports two AMQP connectors:

- zircote/amqp
- ext-amqp

Note: Note that at this time the **ext-amqp** extension does not support *TLS*; for *TLS* support you will be required to utilize the **zircote/amqp** package.

3.1.1 zircote/amqp

Configuration

The configuration of the *zircote/amqp* implementation for **Rhubarb** is comprised of the following key hierarchy:

- **broker**
  - **type**: the broker class name without the namespace
  - **options: the broker specific options**
    - **exchange**: the name of the target exchange
    - **queue: an array of options related to the queue**
      - **name**: the queue name
      - **arguments**: an array of arguments related to the queue, these are AMQP specific and are documented in the *zircote/amqp* library
    - **uri**: the amqp server/cluster uri (it should match your celery worker configuration)
- **result_store**
  - **type**: the result_store class name without the namespace
  - **options: the result_store specific options**
    - **exchange**: the name of the target exchange
  - **uri**: the amqp server/cluster uri (it should match your celery worker configuration)

Note: These options SHOULD match your celery worker configuration.

```
$options = array(
    'broker' => array(
        'type' => 'Amqp',
        'options' => array(
            'exchange' => 'celery',
            'queue' => array(
                'name' => 'celery',
                'arguments' => array(
                    'x-ha-policy' => array('S', 'all')
                )
            ),
            'connection' => 'amqp://guest:guest@localhost:5672/celery'
        )
    ),
    'result_store' => array(
        'type' => 'Amqp',
        'options' => array(
            'exchange' => 'celery',
            'connection' => 'amqp://guest:guest@localhost:5672/celery'
        )
    )
);

$rhubarb = new \Rhubarb\Rhubarb($options);
```

3.1.2 ext-amqp

Configuration

The configuration of the *ext-amqp* implementation for **Rhubarb** is comprised of the following key hierarchy:

- **broker**
  - **type**: the broker class name without the namespace
  - **options: the broker specific options**
    - **exchange**: the name of the target exchange
    - **queue: an array of options related to the queue**
      - **name**: the queue name
      - **arguments**: an array of arguments related to the queue, these are AMQP specific
    - **uri**: the amqp server/cluster uri (it should match your celery worker configuration)
- **result_store**
  - **type**: the result_store class name without the namespace
  - **options: the result_store specific options**
    - **exchange**: the name of the target exchange
  - **uri**: the amqp server/cluster uri (it SHOULD match your celery worker configuration)

Note: These options SHOULD match your celery worker configuration.

```
$options = array(
    'broker' => array(
        'type' => 'PhpAmqp',
        'options' => array(
            'exchange' => 'celery',
            'queue' => array(
                'arguments' => array(
                )
            ),
            'connection' => 'amqp://guest:guest@localhost:5672/celery'
        )
    ),
    'result_store' => array(
        'type' => 'PhpAmqp',
        'options' => array(
            'exchange' => 'celery',
            'connection' => 'amqp://guest:guest@localhost:5672/celery'
        )
    )
);

$rhubarb = new \Rhubarb\Rhubarb($options);
```

3.2 redis

Rhubarb currently supports one **redis connectors:**

- predis/predis

3.2.1 predis/predis

Configuration

The configuration of the *predis/predis* implementation for **Rhubarb** is comprised of the following key hierarchy:

- **broker**
  - **type**: the broker class name without the namespace
  - **options: the broker specific options**
    - **exchange**: the name of the target exchange
    - **uri**: the amqp server/cluster uri (it should match your celery worker configuration)
- **result_store**
  - **type**: the result_store class name without the namespace
  - **options: the result_store specific options**
    - **exchange**: the name of the target exchange
    - **uri**: the amqp server/cluster uri (it should match your celery worker configuration)

Note: These options SHOULD match your celery worker configuration.

```
  $options = array(
      'broker' => array(
          'type' => 'Predis',
          'options' => array(
              'exchange' => 'celery',
              'connection' => 'redis://localhost:6379/1'
          )
      ),
      'result_store' => array(
          'type' => 'Predis',
          'options' => array(
              'exchange' => 'celery',
          )
      )
  );

$rhubarb = new \Rhubarb\Rhubarb($options);
```

3.3 Mongo

Warning: as of Celery v3.1 MongoDB is no longer officially supported. Refer to Rhubarb v.0.2

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*