使用 RabbitMQ 的 PHP

关键点

- 使用“php-amqplib/php-amqplib”版本 2.6.* 似乎需要通过 Composer 安装，并按照基本步骤发送和接收 RabbitMQ 的消息。
- 该过程涉及设置连接、声明队列，并使用 `basic_publish` 方法发送和 `basic_consume` 方法接收，这可能与较新版本有所不同。
- 研究表明，对于版本 2.6.*，消费过程需要手动循环使用 `wait()`，而较新版本可能使用 `consume()` 方法。

安装和设置

要开始使用“php-amqplib/php-amqplib”版本 2.6.*，首先使用 Composer 安装它，运行以下命令：

```
composer require "php-amqplib/php-amqplib:2.6.*"
```

确保 RabbitMQ 已安装并在您的系统上运行，通常可以在 `localhost:5672` 访问，使用默认凭据（`guest/guest`）。如果您的设置不同，请调整这些设置。

发送消息

要发送消息，包括必要的 PHP 文件并创建连接：

```
require_once __DIR__ . '/vendor/autoload.php';
use PhpAmqpLib\Connection\AMQPStreamConnection;
use PhpAmqpLib\Message\AMQPMessage;

$connection = new AMQPStreamConnection('localhost', 5672, 'guest', 'guest');
$channel = $connection->channel();
```

声明队列并发布您的消息：

```
$channel->queue_declare('hello', false, false, false, false);
$msg = new AMQPMessage('Hello World!');
$channel->basic_publish($msg, '', 'hello');
echo " [x] Sent 'Hello World!'\n";
```

最后，关闭连接：

```
$channel->close();
$connection->close();
```

接收消息

接收消息时，设置类似，但定义一个回调函数来处理消息：

```
require_once __DIR__ . '/vendor/autoload.php';
use PhpAmqpLib\Connection\AMQPStreamConnection;

$connection = new AMQPStreamConnection('localhost', 5672, 'guest', 'guest');
$channel = $connection->channel();
$channel->queue_declare('hello', false, false, false, false);

$callback = function ($msg) {
    echo ' [x] Received ', $msg->body, "\n";
};

$channel->basic_consume('hello', '', false, true, false, false, $callback);
while (count($channel->callbacks)) {
    $channel->wait();
}

$channel->close();
$connection->close();
```

请注意，对于版本 2.6.*，您需要使用 `wait()` 循环来保持消费，这与较新版本可能使用更简单的 `consume()` 方法不同。

调查说明：详细使用“php-amqplib/php-amqplib”版本 2.6.*

本节提供了使用“php-amqplib/php-amqplib”库的全面指南，特别是版本 2.6.*，用于与 RabbitMQ 交互，这是一种流行的消息队列系统。信息来自官方文档、教程和版本特定的详细信息，确保开发人员能够全面理解。

背景和上下文 “php-amqplib/php-amqplib”是一个用于与 RabbitMQ 通信的 PHP 库，实现了 AMQP 0.9.1 协议。版本 2.6.* 是较旧的发布版本，尽管该库已经在 2025 年 3 月演变为 3.x.x 版本，但了解其在该特定版本中的使用对于遗留系统或特定项目需求至关重要。该库由包括 Ramūnas Dronga 和 Luke Bakken 在内的贡献者维护，VMware 工程师在 RabbitMQ 上有显著的参与（GitHub - php-amqplib/php-amqplib）。

RabbitMQ 教程，如官方 RabbitMQ 网站上的教程，提供了通常适用但可能反映较新版本的示例。对于版本 2.6.*，需要进行调整，特别是在消费过程中，如下所述。

安装过程 首先，使用 Composer 安装库，这是 PHP 依赖管理器。在项目目录中运行以下命令：

```
composer require "php-amqplib/php-amqplib:2.6.*"
```

此命令确保库被下载并配置以供使用，Composer 管理依赖项。确保 RabbitMQ 已安装并运行，通常可以在 `localhost:5672` 访问，使用默认凭据（`guest/guest`）。对于生产环境，根据需要调整主机、端口和凭据，并参考 CloudAMQP PHP 文档 了解托管代理设置。

发送消息：逐步指南 发送消息涉及建立连接并发布到队列。以下是过程：

1. **包含所需文件：** 使用 Composer 自动加载器包含库：

   ```
   require_once __DIR__ . '/vendor/autoload.php';
   use PhpAmqpLib\Connection\AMQPStreamConnection;
   use PhpAmqpLib\Message\AMQPMessage;
   ```

2. **创建连接和通道：** 初始化与 RabbitMQ 的连接并打开通道：

   ```
   $connection = new AMQPStreamConnection('localhost', 5672, 'guest', 'guest');
   $channel = $connection->channel();
   ```

   参数是主机、端口、用户名和密码，如上所示。对于 SSL 或其他配置，请参考 RabbitMQ PHP 教程。

3. **声明队列并发布：** 声明队列以确保其存在，然后发布消息：

   ```
   $channel->queue_declare('hello', false, false, false, false);
   $msg = new AMQPMessage('Hello World!');
   $channel->basic_publish($msg, '', 'hello');
   echo " [x] Sent 'Hello World!'\n";
   ```

   这里，`queue_declare` 创建一个名为 'hello' 的队列，具有默认设置（非持久、非独占、自动删除）。`basic_publish` 将消息发送到队列。

4. **关闭连接：** 发送后，关闭通道和连接以释放资源：

   ```
   $channel->close();
   $connection->close();
   ```

该过程在版本之间是标准的，变更日志中没有记录版本 2.6.* 与较新版本之间的显著变化。

接收消息：版本特定详细信息 在版本 2.6.* 中接收消息需要仔细注意，因为消费机制与较新版本不同。以下是详细过程：

1. **包含所需文件：** 与发送类似，包含自动加载器和必要的类：

```
require_once __DIR__ . '/vendor/autoload.php';
use PhpAmqpLib\Connection\AMQPStreamConnection;
```

2. **创建连接和通道：**与之前一样建立连接和通道：

```
$connection = new AMQPStreamConnection('localhost', 5672, 'guest', 'guest');
$channel = $connection->channel();
```

3. **声明队列：**确保队列存在，与发送者的声明匹配：

```
$channel->queue_declare('hello', false, false, false, false);
```

4. **定义回调：**创建一个回调函数来处理接收到的消息：

```
$callback = function ($msg) {
    echo ' [x] Received ', $msg->body, "\n";
};
```

   该函数将为每条消息调用，在此示例中打印消息体。

5. **消费消息：**对于版本 2.6.*，使用 `basic_consume` 注册回调，然后进入循环以保持消费：

```
$channel->basic_consume('hello', '', false, true, false, false, $callback);
while (count($channel->callbacks)) {
    $channel->wait();
}
```

   `basic_consume` 方法接受队列名称、消费者标签、no-local、no-ack、独占、no-wait 和回调的参数。循环与 `wait()` 保持消费者运行，检查消息。这是一个重要的细节，因为较新版本（例如 3.2）可能使用 `consume()` 方法，该方法在 2.6.* 中根据 API 文档审查不存在。

6. **关闭连接：**消费后，关闭资源：

```
$channel->close();
$connection->close();
```

意外的细节是版本 2.6.* 中需要手动循环，这可能需要在生产中进行额外的错误处理，例如捕获连接问题的异常。

版本特定考虑 版本 2.6.* 是较旧的发布版本，尽管变更日志中没有明确列出它，但 2.5 到 2.7 之间的版本显示了增强功能，如心跳支持和 PHP 5.3 兼容性。对于大消息，版本 2.6.* 支持在通道上设置 `setBodySizeLimit` 以处理内存限制，如果需要则截断消息，详细信息请参阅 GitHub - php-amqplib/php-amqplib。

与版本 3.2 相比，更改包括对 PHP 8 的支持和新方法如 `consume()`，但发送和基本消费的核心功能保持相似。用户应测试兼容性，特别是与 PHP 版本，因为 2.6.* 可能支持 PHP 5.3 到 7.x，根据变更日志条目。

错误排除和最佳实践

- 如果发送失败，请检查 RabbitMQ 日志中的资源警报，例如磁盘空间低于 50 MB，并通过 RabbitMQ 配置指南 调整设置。
- 对于消费，请确保消费者连续运行；在生产中使用工具如 Supervisor 进行守护进程化。
- 使用 `rabbitmqctl list_queues` 在 Linux 上或 `rabbitmqctl.bat list_queues` 在 Windows 上列出队列，作为特权用户，参考 RabbitMQ 命令行工具。

表：关键方法的版本比较

方法	版本 2.6.* 行为	版本 3.2 行为
`basic_publish`	标准，发布到队列	未更改，相同使用
`basic_consume`	需要使用 `wait()` 循环进行消费	类似，但可用 `consume()` 方法
`setBodySizeLimit`	支持大消息，如果设置则截断	支持，相同功能
PHP 兼容性	可能 5.3 到 7.x	支持 PHP 7.x 和 8.x

该表突出了消费的关键差异，这对于版本 2.6.* 用户至关重要。

结论 使用“php-amqplib/php-amqplib”版本 2.6.* 涉及标准的 Composer 安装，通过 `basic_publish` 发送消息，通过 `basic_consume` 与手动循环接收消息。虽然与较新版本相似，但消费过程需要额外注意，使得开发人员必须测试并调整遗留系统。有关更多详细信息，请参考官方 RabbitMQ 教程和 GitHub 存储库。

关键引用

- GitHub - php-amqplib/php-amqplib 最广泛使用的 PHP 客户端
- RabbitMQ 教程 - “Hello World!” PHP 示例
- CloudAMQP PHP 文档 使用 CloudAMQP 和 PHP 入门
- php-amqplib/php-amqplib Packagist 以前的 videlalvaro/php-amqplib
- GitHub - php-amqplib/php-amqplib CHANGELOG 文件
- RabbitMQ 配置指南 配置项和设置
- RabbitMQ 命令行工具 CLI 参考和使用