
Pro Magento
PRO Magento
Meetup #9
Deep dive into Magento2
queuing functionality

Magento Pro

Зміст

- Що таке Message Brokers
- Опис протоколу AMQP
- RabbitMQ
- Як в Magento 2 реалізована взаємодія с RabbitMQ
- Які core-модулі використовують функціонал черг
- Як додати підтримку черг в Ваш модуль
- Як додати підтримку додаткового message broker в Magento2

Що таке Message Broker

Це програма-посередник, яка використовується для комунікації і обміну інформацією між різними компонентами системи.

Брокери можуть бути використані для:

- валідації
- зберігання
- маршрутизації
- доставки

Переваги використання

- комунікація між різними компонентами системи
- можливість виконувати розрахунки відкладено
- надійність доставки даних між елементами системи

Недоліки використання

- ускладнення системи в цілому
- потенційна поява помилок, які буде важко діагностувати
- витрата часу на вивчення і імплементацію нового програмного компонента в систему

Pro Magento

VS

Magento Pro

Опис AMQP

AMQP – це набір стандартів, котрі регулюють внутрішній процес відправки повідомлень в AMQP-сумісних брокерах.

Magento Pro

Опис AMQP

AMQP-специфікацію можна розділити на:

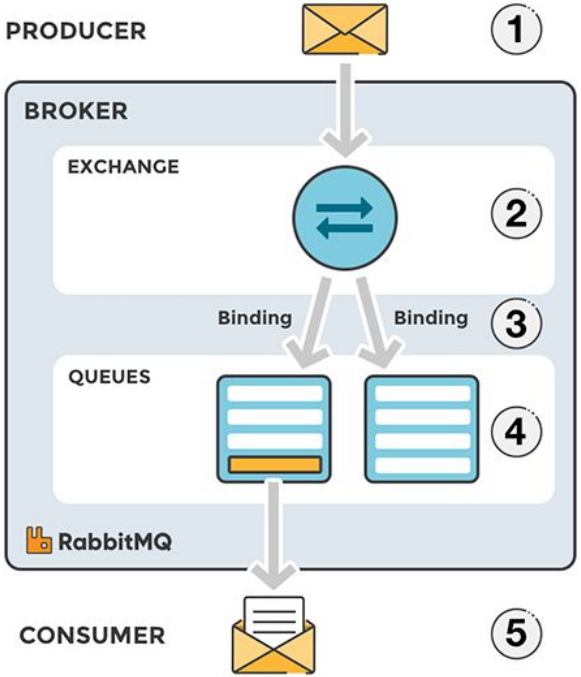
- AMQ-model - application layer level specification
- AMQP - network level protocol

Magento Pro

Компоненти AMQP

- message queue
- exchange
- binding
- message
- connection
- virtual host

Схема відправки повідомлення



Pro Magento

Exchange types

- Direct Exchange
- Topic Exchange
- Fanout Exchange
- Headers Exchange
- Dead letter Exchange (RabbitMQ)

Magento Pro

RabbitMQ

RabbitMQ - це open-source Message Broker, котрий повністю підтримує протокол AMQP. Він реалізований за допомогою plug-in архітектури і окрім AMQP підтримує Streaming Text Oriented Messaging Protocol (STOMP), MQ Telemetry Transport (MQTT).

Magento Pro

Як Magento 2 взаємодіє з RabbitMQ

Низькорівнева функціональність поділена між модулями:

- magento/framework-message-queue
- magento/framework-amqp

Як Magento 2 взаємодіє з RabbitMQ

Модуль magento/framework-message-queue містить реалізацію:

- Publisher
- PublisherPool
- Consumer
- ConnectionTypeResolver
- Envelop
- QueueRepository

Magento Pro

Як Magento 2 взаємодіє з RabbitMQ

Модуль magento/framework-amqp містить функціональність:

- ConnectionFactory
- ConnectionTypeResolver (специфічний для AMQP)
- Exchange
- Queue
- TopologyInstaller

Як Magento 2 взаємодіє з RabbitMQ

Налаштування з’єднання з RabbitMQ:

- при встановленні системи - запуск команди bin/magento setup:install з опціями:
  - amqp-host
  - amqp-user
  - amqp-password
  - amqp-port
  - amqp-virtualhost
  - amqp-ssl
- додати налаштування в файл env.php
- виконати команду bin/magento setup:config:set

Як Magento 2 взаємодіє з RabbitMQ

Налаштування в env.php

```
'queue' =>
 array (
  'amqp' =>
  array (
   'host' => 'rabbitmq.example.com',
   'port' => '11213',
   'user' => 'magento',
   'password' => 'magento',
   'virtualhost' => '/'
  ),
 )
```

Як Magento 2 взаємодіє з RabbitMQ

Конфігураційні файли для налаштування роботи з чергами:

- communication.xml
- queue.xml
- topology.xml
- queue_publisher.xml
- queue_consumer.xml

Як Magento 2 взаємодіє з RabbitMQ

файл communication.xml

```xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Communication/etc/communication.xsd">
    <topic name="async.V1.inventory.bulk-product-source-assign.POST" is_synchronous="false" schema="Magento\InventoryCatalogApi\Api\BulkSourceAssignInterface::execute">
        <handler name="async" type="Magento\InventoryCatalogApi\Api\BulkSourceAssignInterface" method="execute" />
    </topic>
    <topic name="async.V1.inventory.bulk-product-source-unassign.POST" is_synchronous="false" schema="Magento\InventoryCatalogApi\Api\BulkSourceUnassignInterface::execute" >
        <handler name="async" type="Magento\InventoryCatalogApi\Api\BulkSourceUnassignInterface" method="execute" />
    </topic>
    <topic name="async.V1.inventory.bulk-product-source-transfer.POST" is_synchronous="false" schema="Magento\InventoryCatalogApi\Api\BulkInventoryTransferInterface::execute"
        <handler name="async" type="Magento\InventoryCatalogApi\Api\BulkInventoryTransferInterface" method="execute" />
    </topic>
    <topic name="inventory.source.items.cleanup" is_synchronous="false" request="string[]">
        <handler name="inventory.source.items.handler" type="Magento\InventoryCatalog\Model\DeleteSourceItemsBySkus" method="execute" />
    </topic>
    <topic name="inventory.mass.update" is_synchronous="false" request="Magento\InventoryCatalog\Model\UpdateInventory\InventoryData">
        <handler name="inventory.mass.update" type="Magento\InventoryCatalog\Model\UpdateInventory" method="execute" />
    </topic>
</config>
```

Magento Pro

Як Magento 2 взаємодіє з RabbitMQ

файл queue_consumer.xml

```xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework-message-queue:etc/consumer.xsd">
    <consumer name="inventory.source.items.cleanup" queue="inventory.source.items.cleanup" connection="db"
              handler="Magento\InventoryCatalog\Model\DeleteSourceItemsBySkus::execute" />
    <consumer name="inventory.mass.update" queue="inventory.mass.update" connection="db"
              handler="Magento\InventoryCatalog\Model\UpdateInventory::execute" />
</config>
```

Як Magento 2 взаємодіє з RabbitMQ

файл queue_topology.xml

```xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework-message-queue:etc/topology.xsd">
    <exchange name="magento-db" type="topic" connection="db">
        <binding id="InventorySourceItemsCleanUpBinding" topic="inventory.source.items.cleanup"
                 destinationType="queue" destination="inventory.source.items.cleanup"/>
        <binding id="InventoryMassUpdateBinding" topic="inventory.mass.update" destinationType="queue"
                 destination="inventory.mass.update"/>
    </exchange>
</config>
```

Як Magento 2 взаємодіє з RabbitMQ

файл queue_publisher.xml

```xml
<?xml version="1.0"?>
<!--
/**
 * Copyright © Magento, Inc. All rights reserved.
 * See COPYING.txt for license details.
 */
-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework-messa
    <publisher topic="async.V1.inventory.bulk-product-source-assign.POST">
        <connection name="amqp" exchange="magento" disabled="false" />
    </publisher>
    <publisher topic="async.V1.inventory.bulk-product-source-unassign.POST">
        <connection name="amqp" exchange="magento" disabled="false" />
    </publisher>
    <publisher topic="async.V1.inventory.bulk-product-source-transfer.POST">
        <connection name="amqp" exchange="magento" disabled="false" />
    </publisher>
    <publisher topic="inventory.source.items.cleanup">
        <connection name="db" exchange="magento-db" disabled="false" />
    </publisher>
    <publisher topic="inventory.mass.update">
        <connection name="db" exchange="magento-db" disabled="false" />
    </publisher>
</config>
```

Як Magento 2 взаємодіє з RabbitMQ

Magento\Framework\MessageQueue\PublisherInterface
(\Magento\Framework\MessageQueue\PublisherPool)

\Magento\Framework\MessageQueue\Publisher

\Magento\Framework\MessageQueue\ExchangeRepository

\Magento\Framework\Amqp\Exchange

\Magento\Framework\Amqp\Exchange::enqueue -> AMQPMessage -> AMQPChannel ($message, $exchange, $topic)

Magento Pro

Використання черг в core-модулях

- magento/module-product-alert
- magento/module-webapi-async

Magento Pro

Функціонал черг в кастомному модулі

Додамо в наш новий модуль наступні конфігураційні файли:

- communication.xml
- queue_topology.xml
- queue_consumer.xml
- queue_publisher.xml

Функціонал черг в кастомному модулі

Визначимо клас-publisher, який буде використовувати стандартний \Magento\Framework\MessageQueue\PublisherInterface для публікації повідомлень

```php
<?php
declare(strict_types=1);

namespace ITDelight\ExampleQueue\Model\MessageQueues;

use Magento\Framework\MessageQueue\PublisherInterface;

class Publisher
{
    private const QUEUE_TOPIC = 'it_delight_example_queue';

    private PublisherInterface $publisher;

    public function __construct(PublisherInterface $publisher)
    {
        $this->publisher = $publisher;
    }

    public function publish(string $messageData): void
    {
        $this->publisher->publish( topicName: self::QUEUE_TOPIC, $messageData);
    }
}
```

Функціонал черг в кастомному модулі

Додамо також клас-consumer, який буде опрацьовувати інформацію з черги

```php
<?php
declare(strict_types=1);

namespace ITDelight\ExampleQueue\Model\MessageQueues;

use Psr\Log\LoggerInterface;

class Consumer
{
    private LoggerInterface $logger;

    public function __construct(LoggerInterface $logger)
    {
        $this->logger = $logger;
    }

    public function processMessage(string $message): void
    {
        $this->logger->info( message: 'Example queue handler output: ' . $message);
    }
}
```

Pro Magento

Redis як Message Broker

1. Створити новий модуль
2. Створити redis-client
3. Створити Publisher, ConnectionTypeResolver, Exchange специфічні для Redis
4. Імплементувати підтримку redis в message-queue архітектуру Magento2