**Super.br**

**Lab 05 - Backend - Utilizando mensageria**

1. Instalação dos pacotes

Rode os comando abaixo para criar instalar o pacote:

|  |
| --- |
| **composer require symfony/messenger** |

Instale o transportador que será usado, RabbitMQ:

|  |
| --- |
| **composer require symfony/doctrine-messenger** |

Atualize o schema do banco:

php bin/console doctrine:schema:update --force

2. Manipulando mensagens

Crie um aquivo de mensagem src/Message/SmsNotification.php:

<?php

// src/Message/SmsNotification.php

namespace App\Message;

class SmsNotification

{

private $content;

public function \_\_construct(string $content)

{

$this->content = $content;

}

public function getContent(): string

{

return $this->content;

}

}

Crie um manipulador de mensagem:

<?php  
// src/MessageHandler/SmsNotificationHandler.php  
namespace App\MessageHandler;  
use App\Message\SmsNotification;  
use Symfony\Component\Messenger\Attribute\AsMessageHandler;  
  
#[AsMessageHandler]  
class SmsNotificationHandler  
{  
 public function \_\_invoke(SmsNotification $message)  
 {  
 *// ... do some work - like sending an SMS message!*  
}  
}

Execute o comando para conferir se os manipuladores estão corretos:

|  |
| --- |
| **php bin/console debug:messenger** |

Para despachar a mensagem, crie um controlador chamado MensagemController, injete o messenger.default\_bus serviço (através do MessageBusInterface):

<?php  
  
namespace App\Controller;  
use App\Message\SmsNotification;  
use Symfony\Component\Routing\Annotation\Route;  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\Messenger\MessageBusInterface;  
use Symfony\Component\HttpFoundation\Response;  
  
class MensagemController extends AbstractController  
{  
  
 #[Route('/sms')]  
 public function index(MessageBusInterface $bus): Response  
 {  
  
 *// will cause the SmsNotificationHandler to be called*   
  
$bus->dispatch(new SmsNotification('Teste de mensagem!'));  
  
 return new Response(  
 '<html><body>Sucesso</body></html>'  
 );  
  
 }  
  
}   
  
Transportes para mensagem assíncronas

Por padrão, as mensagens são tratadas assim que são despachadas. Se você deseja manipular uma mensagem de forma assíncrona, pode configurar um transporte. Um transporte é capaz de enviar mensagens (por exemplo, para um sistema de filas) e depois recebê-las por meio de um trabalhador . O Messenger suporta vários transportes .

Instale o pacote messenger-amqp: composer require symfony/amqp-messenger

No arquivo .env descomente a linha que utiliza o doctrine como transporte:

MESSENGER\_TRANSPORT\_DSN=doctrine://default?auto\_setup=0

A seguir, em config/packages/messenger.yaml, vamos definir um transporte chamado async que usa esta configuração:

# config/packages/messenger.yaml

framework:

messenger:

transports:

async: "%env(MESSENGER\_TRANSPORT\_DSN)%"

# or expanded to configure more options

#async:

# dsn: "%env(MESSENGER\_TRANSPORT\_DSN)%"

# options: []

Agora que você tem um transporte configurado, em vez de tratar uma mensagem imediatamente, você pode configurá-los para serem enviados a um transporte:

# config/packages/messenger.yaml

framework:

messenger:

transports:

async: "%env(MESSENGER\_TRANSPORT\_DSN)%"

routing:

# async is whatever name you gave your transport above

'App\Message\SmsNotification': async

Graças a isso, o App\Message\SmsNotificationserá enviado para o async transporte e seu(s) handler(s) não será(ão) chamado(s) imediatamente. Quaisquer mensagens não correspondidas routing ainda serão tratadas imediatamente, ou seja, de forma síncrona.

Faça a chamada novamente ao controller criado, depois disso execute o comando para verificar a fila:

**php bin/console dbal:run-sql 'SELECT \* FROM messenger\_messages'**

Execute o comando:

**php bin/console messenger:consume async**

Veja que a fila será consumida. Execute o select novamente e verá que a fila não existe mais no banco de dados:

**php bin/console dbal:run-sql 'SELECT \* FROM messenger\_messages'**

Exercício 1

Utilizando o conceito das filas, crie uma classe de mensagem e seu respetivo manipulador para salvar Processo e Documentos no banco de dados.

Modifique os Controllers respectivos para que ao salvar um novo registro no formulário, ele crie uma fila para execução assíncrona dos jobs.

OBS: Dúvidas acesse a documentação:

**https://symfony.com/doc/current/messenger.html**