cloud-campus

FORMATION SYMFONY

PROMO 4

Pascal WACHE

Software Engineer | Data Scientist

nguessiandre@yahoo.fr

Objectifs

- Quelle est la structure de Symfony ?
- Qu'est ce qu'un évènement Symfony ?
- Quels sont les types d'évènement Symfony ?
- Comment gérer un évènement sur Symfony ?
- Comment gérer les évènements asynchrones sur Symfony ?
- Pratiquer toutes ces notions autour d'un projet.

Plan du cours

I. Symfony
   I. Objectif
   II. Structure
II. Les évènements
   I. Principe
   II. EventListener
   III. EventSubscriber
   IV. Doctrine Events
   V. Symfony Asynchronous
III. Cas pratique
   I. Déroulement d'un projet
   II. Discussion sur les cas d'usage

Symfony

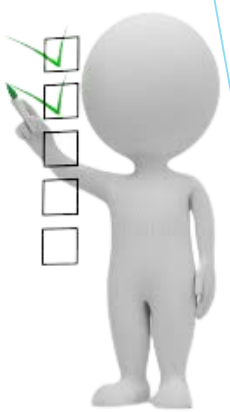
Définition et structure

Symfony

- Approche traditionnelle

Spécifications

Implémentation



Validation

Optimisation du processus

Time

framework

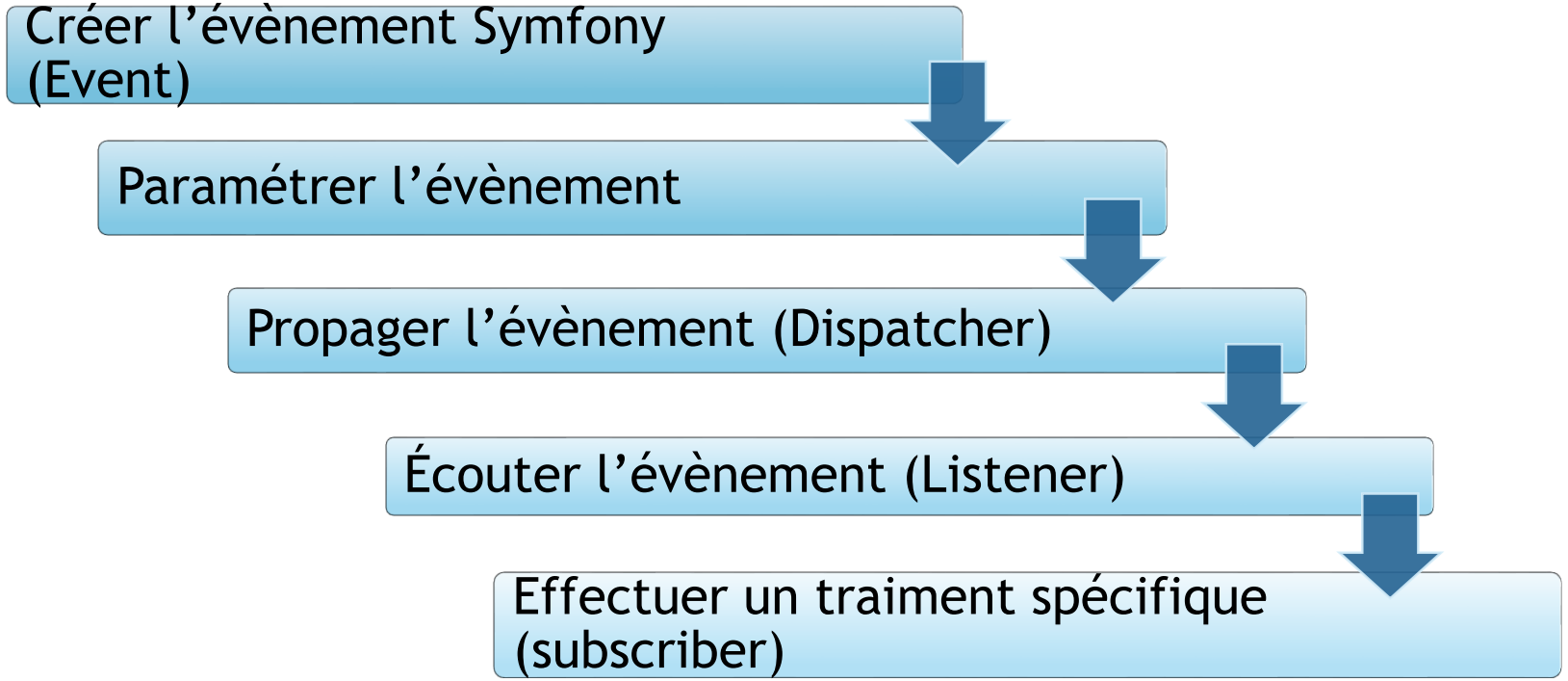
CMS

WebFlow

Symfony

- **Framework** pour le développement des applications web, basé sur le langage PHP

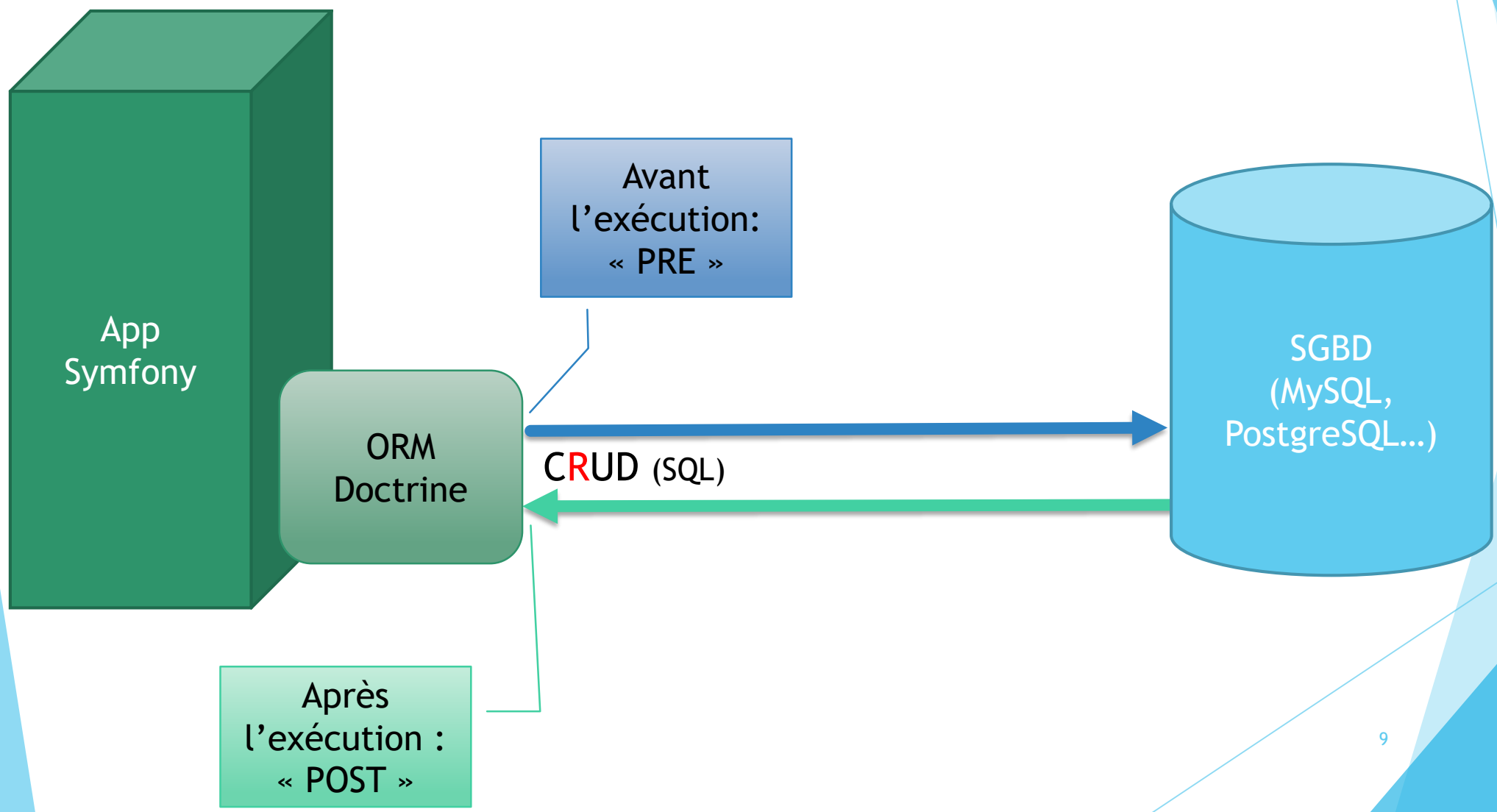
Les évènements



Créer un Event Symfony

- Class extends ***Symfony\Contracts\EventDispatcher\Event;***

Les évènements Doctrine



Les évènements Doctrine

Action	Avant	Après
Persist	prePersist	postPersist
Update	preUpdate	postUpdate
Delete	preRemove	postRemove

Gérer les évènements :

- Avec un Listener
- Dans l'objet directement

Doctrine ORM – Event in Listener

```php
namespace App\EventListener;

use App\Entity\Contact;
use Doctrine\ORM\Event\LifecycleEventArgs;
use Symfony\Contracts\EventDispatcher\Event;

class ContactListener{
    public function prePersist(LifecycleEventArgs $args):void    {
        $entity = $args->getEntity();
        if($entity instanceof Contact){
            $entity->setStatus($entity->getStatus()+2);
        }
    }
}
```

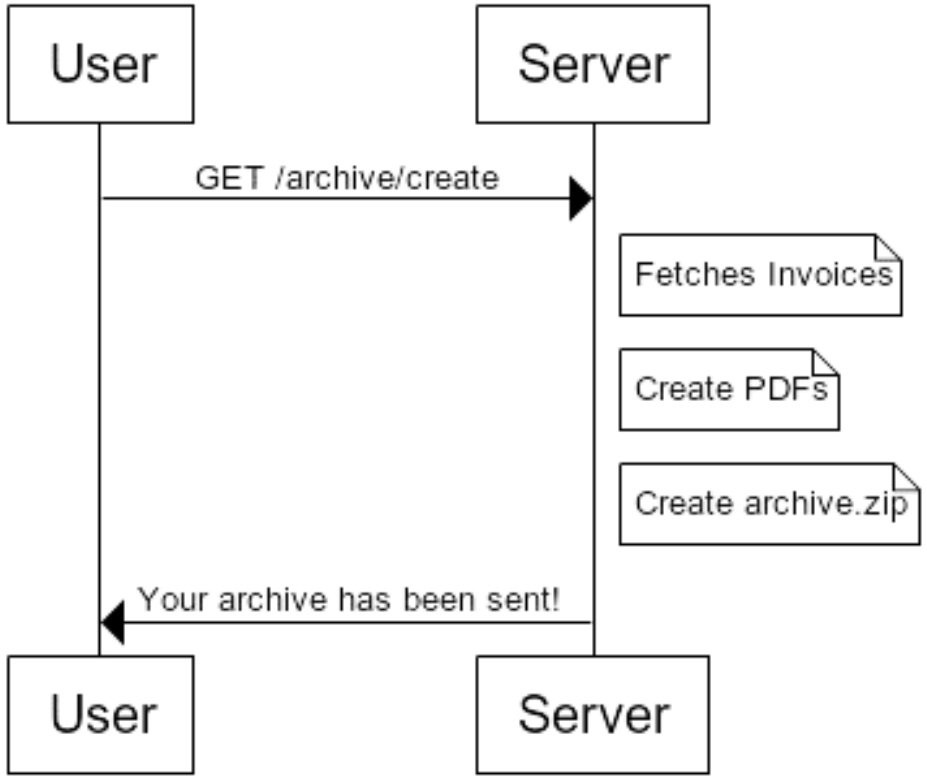
Doctrine ORM – Event in File

```
…
#[ORM\Entity(repositoryClass: ContactRepository::class)]
class Contact
{    …
    #[ORM\Column(length: 255)]
    private ?string $name = null;
    #[ORM\PrePersist]
    public function beforePersist(){
        $this->status = $this->status+2;
    }
    #[ORM\PostPersist]
    public function afterPersist(){
        $this->status = $this->status+2;
    }
}
```

Symfomy Asynchrone

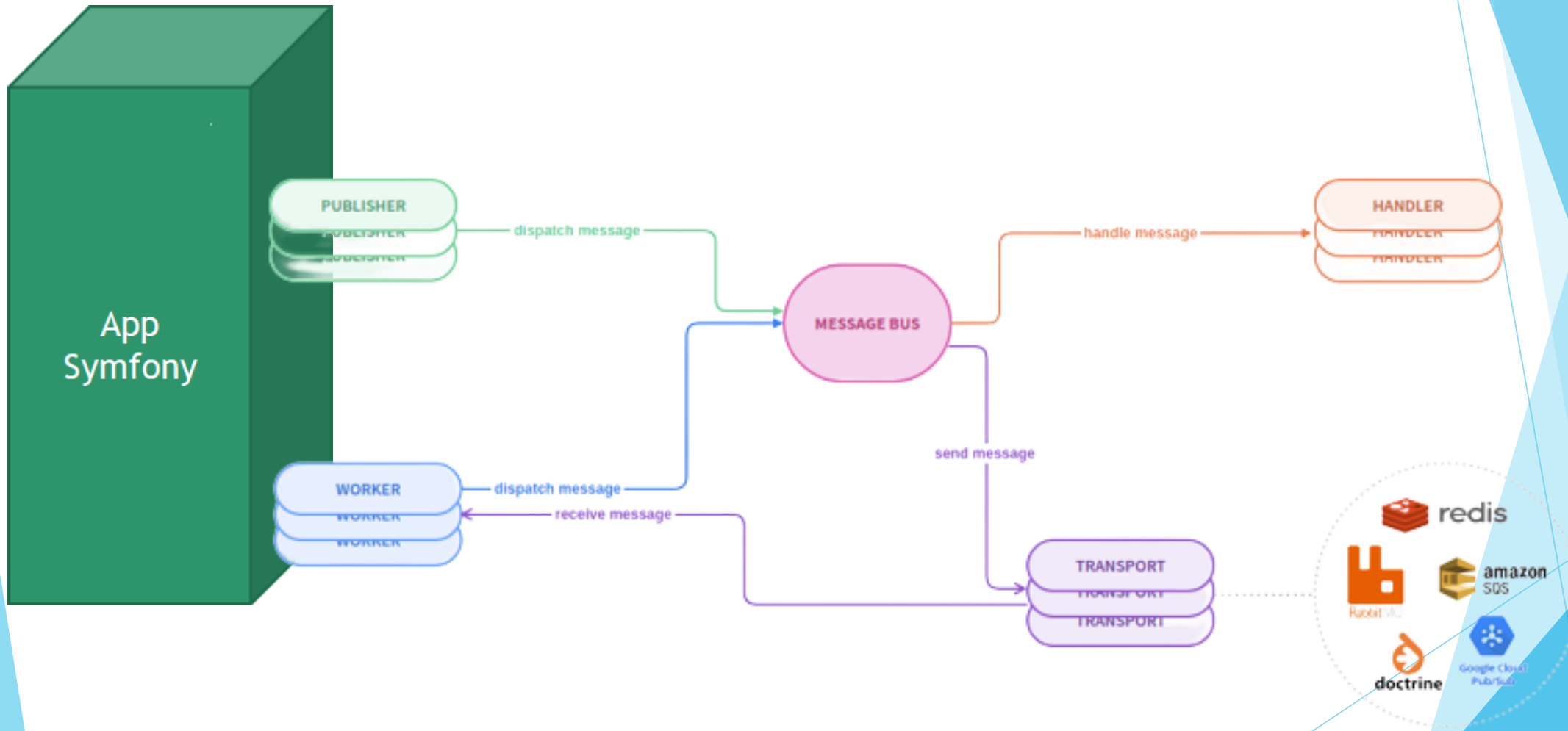
- Problématique
- Solutions existantes
- Symfony Messenger
- RabbitMq
- * Profiler Symfony

Symfomy Asynchrone



Illustration

- Pour des actions qui s'exécutent en arrière plan
- Permet d'améliorer l'expérience utilisateur
- Réduis les délais d'inactivité utilisateur
- Rend l'application plus robuste et souple


App
Symfony
PUBLISHER
dispatch message
MESSAGE BUS
handle message
HANDLER
WORKER
dispatch message
receive message
send message
TRANSPORT
redis
amazon
SQS
doctrine

Symfomy Asynchrone

Message

```
…
class MyComplexActionMessage
{
        private $param1; …
        public function __construct(int $param1){
                $this->param1 = $param1; …
        }
        public function getParam1() {
                return $this->param1;
        }
}
```

Symfony Messenger

MessageHandler (Worker)

```
use Symfony\Component\Messenger\Handler\MessageHandlerInterface;

class MyComplexActionHandler implements MessageHandlerInterface
{
    // Utiliser l'autoinjection des dependances de vos actions complexes

    public function __invoke(MyComplexActionMessage $message)
    {
        $param1 = $message->getParam1(); // Ajouter des traitements supplementaires ici
        $this->complexAction($param1);
    }
    public function complexAction1($param1) {
        // Complex code here
    }
}
```

Symfomy Asynchrone

Dispatcher

```
use Symfony\Component\Messenger\MessageBusInterface;

class ComplexActionController extends AbstractController
{
    public function index(MessageBusInterface $bus)
    {
        ...
        $bus->dispatch(new ComplexActionMessage($param1));
    }
}
```

Symfomy Asynchrone

Transport

```yaml
framework:
    messenger:
        transports:
            async:
                dsn: '%env(MESSENGER_TRANSPORT_DSN)%'
                retry_strategy:
                    max_retries: 3
                    multiplier: 2
            failed: 'doctrine://default?queue_name=failed'
         failure_transport: failed
        routing:
            MyComplexActionMessage: async
```

Gestionnaire de Message

Doctrine

Système par defaut utilize par Symfony. Ici les messages sont stockés dans votre base de données

- **`Symfony console debug:messenger`**

  Liste les messages gérer par défaut

- **`Symfony messenger:consume async -vv`**

  Lance un processus de traitement des messages dans la file d'attente

- **`Symfony console messenger:failed:show`**

  Récupère les messages qui ont échoués

- **`Symfony console messenger:failed:retry`**

  Relance l'exécution des messages qui ont échoués

Gestionnaire de Message

Capturer les erreurs

- *Implémenter l'interface un **EventSubscriverInterface***
- *Implémenter un Listener sur l'évènement **WorkerMessageFailedEvent***
- *Pour accéder au message échoué :*
  - *Utiliser les méthodes **WorkerMessageFailedEvent::getEenvelope()->getMessage()***
  - *Utiliser la console : **Symfony console messenger:failed:show()***
  - *Consulter la base de données*

Symfony - Tests

- **Tester permet de vérifier le bon fonctionnement des composants applicatifs de Symfony**

Quelques Concepts

- **Behavior Driven Development (BDD)**

  Définition des users story

- **Domain Driven Development (DDD)**

  Développement orienté par le métier

- **Test Driven Development (TDD)**

  Définition des critères d'évaluation de l'application avent de la développer

Types de tests

- **Test unitaires**

  Vérifier le traitement d'une fonction précise

- **Test fonctionnels**

  Vérifier le fonctionnement d'une fonctionnalité

- **Tests de validation (ou End to End)**

  Vérifie de bout en bout le déroulement d'un scénario utilisateur

- **TestCase**

  Test unitaires

- **KernelTestCase**

  Test fonctionnels

- **WebTestCase**

  Test des controllers (les réponses et vues Twig)

Test Symfony avec PHPUnit

Configuration de l'environnement de Test: env.test

- **php bin/console –-env=test doctrine:database:create**
- **php bin/console --env=test doctrine:schema:create**
- **Symfony console make:test**

Crée une classe de test dans le namespace **App\Tests**

- Par convention, la classe se termine par **Test**

ex: **ContactTest**

- Chaque méthode à tester doit commencer par la mot clé ***test***

ex: ***testValidationStatus()***

Crédits

André Pascal WACHE NGUESSI

Data Scientist & Software Engineer

Contact :

nguessiandre@yahoo.fr

https://github.com/Anderson05

linkedin.com/in/andré-pascal-wache-nguessi-92a9b5109