Créer un nouveau Worker

Last edited by **Maelan LE BORGNE** 4 years ago

Voici les étapes à suivre pour développer un nouveau worker dans l'environnement ePack V4

Créer les ressources RabbitMQ

Selon les besoins, définir :

- **Un exchange**
  Créer 1 exchange par fonctionnalité ou worker permet de faciliter la future cohabitation avec ePack V5.
  Définir la propriété x-dead-letter-exchange à ce même exchange permet de profiter de la fonctionnalité "réessayer n fois puis abandonner" sur l'API en définissant la propriété *max_retry* du worker.
- **Un "dead-letter-exchange"**
  Si besoin de router les messages rejeté vers une queue différente. Peux être utile pour créer un délai avant une remise en queue (voir fonctionnement du worker Multisite)
- **Une queue**

Créer le worker

- Créer un classe dans app/tasks/Worker qui hérite de WorkerTask.
- Définir la fonction `initialize()` qui appelle la fonction parente et défini les propriété du worker.
  Exemple :

```
public function initialize()
{
    parent::initialize();
    $this->logger->setMainLogger("ma_fonctionalite");
    $this->TTL = 300;
    $this->setQueue("ma_queue");
}
```

- Définir la logique dans la fonction `callback()`
  Exemple :

```
public function callback(\QueueAdapter\MessageAdapter $message)
    {
        try {
            $body = $message->getBody();
            // Faire quelque chose avec le message
            ...
            $message->acknowledge();
        } catch (Throwable $e){
            //Traiter l'erreur
            ...
            $message->reject();
        }
    }
```

Configurer un logger dédié

Dans la partie précédente, on à défini le logger utilisé par le logger avec la l'instruction `$this->logger->setMainLogger("ma_fonctionalite");` Ce logger n'existe actuellement pas et les logs finiront par défaut dans le logger "error".
Pour configurer un nouveau logger, ajouter un handler et un logger dans le fichier app/config/logger_config.yaml en prenant soin de définir "error_file" comme étant le premier handler du nouveau logger (pour la remonté en bulle des erreurs).

Créer un fichier de configuration Supervisor et le placer dans `supervisord.conf.d`

Exemple :

```
[program:MaFonctionaliteWorker]
command=php -q /opt/epack/api/app/cli.php ma-fonctionalite
process_name=%(program_name)s.%(process_num)s
numprocs=1
directory=/opt/epack/api
stdout_logfile=/var/log/supervisor/%(program_name)s.%(process_num)s.log
```

```
stderr_logfile=/var/log/supervisor/%(program_name)s.%(process_num)s.log
autostart=true
autorestart=true
user=nginx
exitcodes=0
stopsignal=KILL
```

Notez que l'argument de la commander cli.php prend le nom de la classe worker en kebab-case sans "worker" (MaFonctionaliteWorker => ma-fonctionalite) Ajuster la variable numprocs (qui défini le nombre de worker a s'exécuter en simultané) selon le besoin