Nek

Zeste de savoir

Configurez des locks facilement avec Symfony

2 août 2020

Table des matières

La documentation de Symfony décrit très bien comment les locks fonctionnent, il n'est donc pas question dans cet article de faire tout un tuto à propos des locks. Mais en revanche cette même configuration n'explique pas comment configurer très simplement un lock à l'aide de Symfony, alors que c'est possible!

Voyons cela ensemble.

1. Installer le composant lock dans votre application

Rien de plus simple, il suffit de taper la commande suivante dans votre terminal :

```
composer require lock
```

2. Configurer un lock

C'est là où ça se corse. Avec Symfony on a l'habitude d'avoir tout de suite une configuration de base. Mais ce composant n'a pas de recette et n'est donc pas configuré automatiquement par Symfony Flex.

Cependant sa configuration est très simple, voyez vous même:

```
# Dans le fichier packages/lock.yaml
framework:
    lock: 'redis://localhost'
```

Dans cet exemple, nous configurons un lock basé sur la base de données redis. Mais cela pourrait très bien fonctionner avec la base de données que vous utilisez avec doctrine! Il faudrait spécifier `lock: '%env(DATABASE_URL)%'`.

3. Utilisons notre lock

Comme je l'ai spécifié en introduction, cet article n'a pas pour but d'être une explication complète sur les locks, je vais donc en faire une utilisation simple pour la démonstration.

Cette commande fonctionnera toute seule car elle est chargée automatiquement (autowired) par Symfony, Symfony lui injectera donc le lock automatiquement.

```
class LockTestCommand extends Command
{
    protected static $defaultName = 'app:lock-test';

    private $lock;

    public function __construct(LockInterface $lock)
    {
        $this->lock = $lock;
        parent::__construct();
    }

    protected function configure()
    {
        $this
            ->setDescription('This is a simple lock test')
        ;
    }

    protected function execute(InputInterface $input,
    OutputInterface $output): int
    {
        $io = new SymfonyStyle($input, $output);

        while (!$this->lock->acquire()) {
            $io->comment('Je ne peux pas travailler, je suis en
    attente.');
            sleep (1);
        }

        $io->success("Je peux travailler, le travail précédent
    semble terminé !");
        $this->task();

        $this->lock->release();

        return 0;
    }

    private function task()
    {
```

```
39          // On simule une tâche qui prend 10 secondes
40          sleep(10);
41      }
42  }
```

4. Allons un peu plus loin

Dans la configuration que j'ai montré précédemment, nous enregistrons seulement le lock `default`, il sera automatiquement injecté à nos services qui réclament un lock. Mais on peut en réalité définir plusieurs locks (et également conserver celui par défaut pour simplifier les choses dans le cas général). Voici comment faire:

```yaml
framework:
    lock:
        default: '%env(DATABASE_URL)%'
        redis_high_availability: ['redis://r1.docker',
            'redis://r2.docker']
```

Cela va générer un service nommé `lock.redis_high_availability`, et vous devez le spécifier explicitement dans la configuration de vos services si vous souhaitez l'injecter à l'un deux.

La configuration ici présente qui combine deux instances de redis exploite le `Combined Lock` et est comme son nom l'indique tout indiqué dans le cas d'une infrastructure à haute disponibilité.

Notez que pour vos locks, il est préférable d'utiliser des locks nommés. Vous pouvez de la même façon n'injecter que la factory, avoir un lock par resource fait généralement plus de sens:

```php
public function __constructor(LockFactory $lockFactory)
{
    $this->lockFactory = $lockFactory;
}

public function execute(MyEntity $item)
{
    // Votre code...
    $lock = $this->lockFactory->create('item'.$item->getId());
    // Utilisation du lock...
}
```

4. Allons un peu plus loin

J'espère avoir pu en aider plus d'un! 🍊