



Formation PHP 7 Symfony 4.3

Hajer ALAYA

Hajer.adahmeni@gmail.com

Du 15/07/2019 Au 19/07/2019



Plan

- Notions avancées
 - Repository et Profiler
 - Les formulaires et leur validation

Repository

- un dépôt ou référentiel (de l'anglais repository) est un stockage centralisé et organisé de données.
- Utilisation de langage DQL pour les requêtes.
- Exemple :

Soit à afficher à l'utilisateur les dernières offres actives sur la page d'accueil.

Un emploi est considéré actif s'il a été posté il y a moins de 30 jours

Etapes :

1. Ajouter la classe JobRepository
2. Mapping de la classe JobRepository
3. Modifier le méthode index au niveau du JobControleur
4. Lancer des fixtures
5. Démarrer votre serveur web et lancer <http://localhost:8000/>:

Repository

```
<?php // src/Repository/JobRepository.php
namespace App\Repository;
use DateTime;
use Doctrine\ORM\EntityRepository;
use App\Entity\Category;
class JobRepository extends EntityRepository
{ public function findActive(DateTime $date)
  {
    return $this->createQueryBuilder('j')
      ->andWhere('j.createdAt > :date')
      ->setParameter('date', $date)
      ->getQuery()
      ->getResult();
  }
}
```

Repository

- Modifier la configuration du mapping de l'entité Job pour indiquer à Doctrine la classe que l'ORM devra utiliser pour accéder aux données. Cela se passe dans le fichier `config/doctrine/mapping/Job.orm.yml`.

```
# config/doctrine/mapping/Job.orm.yml  
App\Entity\Job:  
    type: entity  
    repositoryClass: App\Repository\JobRepository  
  
# ...debut du fichier
```

Repository

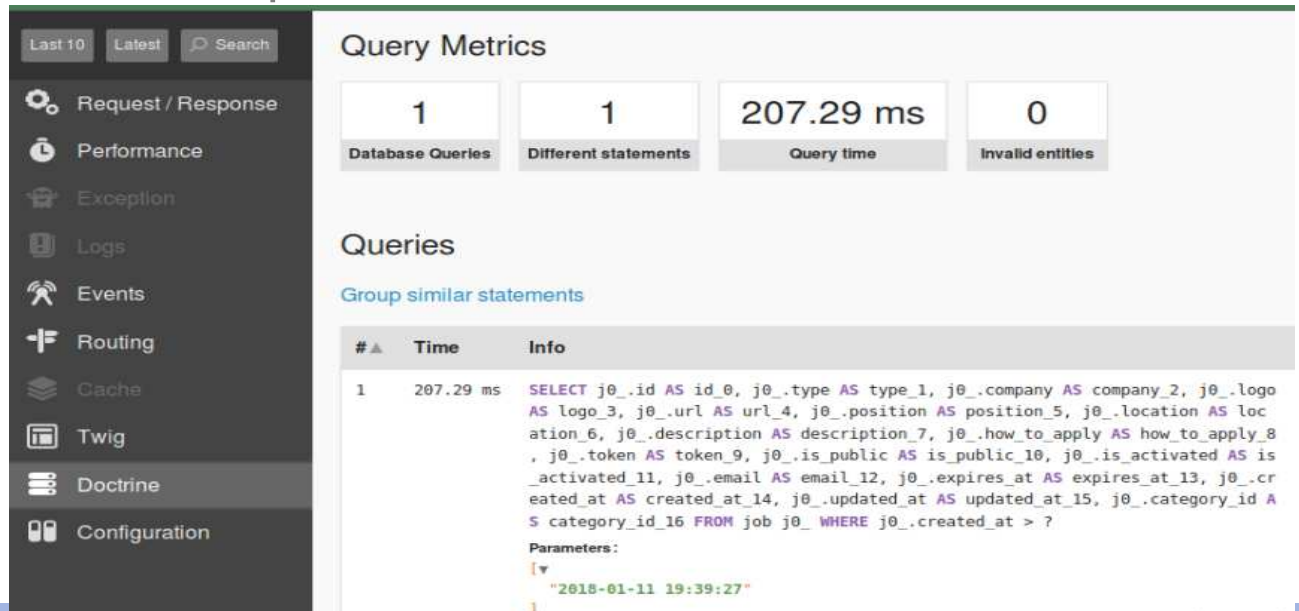
```
use DateTime
Symfony\Component\HttpFoundation\Response
<?php // src/Controller/JobController.php
namespace App\Controller;
// ...
class JobController extends AbstractController
{
    public function index(EntityManagerInterface $em): Response
    {
        $jobs = $em->getRepository(Job::class)->findActive(new DateTime('-30 day'));

        return $this->render('job/index.html.twig', [
            'jobs' => $jobs,
        ]);
    }
}
```

Profiler

- Consulter la requête avec le Profiler
- voir plus de détail des informations récupérées

`composer require symfony/profiler-pack --dev`



The screenshot displays the Symfony Profiler interface. On the left is a sidebar with navigation links: Request / Response, Performance, Exception, Logs, Events, Routing, Cache, Twig, Doctrine, and Configuration. The main content area is titled 'Query Metrics' and shows four statistics: 1 Database Queries, 1 Different statements, 207.29 ms Query time, and 0 Invalid entities. Below this is a 'Queries' section with a link 'Group similar statements'. A table lists the queries, with the first query showing a time of 207.29 ms. The query is a complex SELECT statement with many aliases. The parameters section shows a single parameter: '2018-01-11 19:39:27'.

#	Time	Info
1	207.29 ms	<pre>SELECT j0_.id AS id_0, j0_.type AS type_1, j0_.company AS company_2, j0_.logo AS logo_3, j0_.url AS url_4, j0_.position AS position_5, j0_.location AS location_6, j0_.description AS description_7, j0_.how_to_apply AS how_to_apply_8, j0_.token AS token_9, j0_.is_public AS is_public_10, j0_.is_activated AS is_activated_11, j0_.email AS email_12, j0_.expires_at AS expires_at_13, j0_.created_at AS created_at_14, j0_.updated_at AS updated_at_15, j0_.category_id AS category_id_16 FROM job j0_ WHERE j0_.created_at > ?</pre> <p>Parameters:</p> <pre>[{"2018-01-11 19:39:27"}]</pre>

Repository

- Modifier [AppFixtures](#)
- Recharger les fixtures au travers de la commande:

```
bin/console doctrine:fixtures:load
```

- Démarrer votre serveur web et lancer <http://localhost:8000/>:

```
php -S 127.0.0.1:8000 -t public
```

- *Soit à afficher à l'utilisateur les dernières offres par catégories*
Etapes :
 1. Ajouter la classe CatégorieRepository
 2. Mapping de la classe CategoryRepository
 3. Modifier le méthode index au niveau du JobControleur
 4. Modifier la page twig index.html.twig
 5. Démarrer votre serveur web et lancer <http://localhost:8000/>:

```
<?php // src/Entity/CategoryRepository.php
```

```
declare(strict_types=1);
```

```
namespace App\Repository;
```

```
use DateTime;
```

```
use Doctrine\ORM\EntityRepository;
```

```
class CategoryRepository extends EntityRepository
```

```
{
```

```
    public function findCategoriesWithJobs()
```

```
    {
```

```
        return $this->createQueryBuilder('c')
```

```
            ->join('c.jobs', 'j')
```

```
            ->where('j.expiresAt >= :date')
```

```
            ->setParameter('date', new DateTime())
```

```
            ->getQuery()
```

```
            ->getResult();
```

```
    }
```

```
}
```

- Modifier la configuration du mapping de l'entité Catégorie pour indiquer à Doctrine la classe que l'ORM devra utiliser pour accéder aux données. Cela se passe dans le fichier `config/doctrine/mapping/Category.orm.yml`.

```
# config/doctrine/mapping/Category.orm.yml  
App\Entity\Category:  
    type: entity  
    repositoryClass: App\Repository\CategoryRepository  
  
# ...debut du fichier
```

Repository et Profiler



- Modifier la méthode index dans la classe JobController.

```
public function index(EntityManagerInterface $em): Response
{
    $categories = $em->getRepository(Category::class)->findCategoriesWithJobs();

    $jobsCategories = [];
    foreach ($categories as $category) {
        $jobsCategories[$category->getName()] = $em->getRepository(Job::class)-
>findActiveByCategory($category);
    }
    return $this->render('job/index.html.twig', [
        'categories' => $jobsCategories,
    ]); }
```

Repository et Profiler



- Ajouter la méthode permettant de récupérer les jobs par catégorie dans la classe JobRepository .

```
use App\Entity\Category;
public function findActiveByCategory(Category $category)
{
    return $this->createQueryBuilder('j')
        ->where('j.category = :category')
        ->andWhere('j.expiresAt >= :date')
        ->setParameter('category', $category)
        ->setParameter('date', new DateTime())
        ->getQuery()
        ->getResult();
}
```

- Modifier la tempate twig index.html.twig.

```
{% extends "base.html.twig" %}
{% block body %}
    <h1 class="my-4">Liste des offres</h1>
    {% for category, jobs in categories %}
        <div class="row">
            <h2 style="font-weight: bold; margin: 2rem 0;">{{ category }}</h2>
            {% for job in jobs %}
                <div class="row">
                    <div class="col-md-7">
                        <a href="#">
                            
                        </a>
                    </div>
                    <div class="col-md-5">
                        <h3>{{ job.position }}</h3>
                        <p>{{ job.description }}</p>
                        <p>Posted on {{ job.createdAt|date("m/d/Y") }}</p>
                        <a class="btn btn-primary" href="{{ path('job_show', { 'id': job.id, 'company': job.companySlug, 'location': job.locationSlug, 'position': job.positionSlug }) }}">See more</a>
                    </div>
                </div>
            </div>
            <hr>
        {% endfor %}
    </div>
{% endfor %}
```

- Modifier la template twig index.html.twig.

```
{% extends "base.html.twig" %}
```

```
<ul class="pagination justify-content-center">  
  <li class="page-item disabled">  
    <a class="page-link" href="#" aria-label="Previous">  
      <span aria-hidden="true">&laquo;</span>  
      <span class="sr-only">Previous</span>  
    </a>  
  </li>  
  <li class="page-item">  
    <a class="page-link" href="#">1</a>  
  </li>  
  <li class="page-item disabled">  
    <a class="page-link" href="#" aria-label="Next">  
      <span aria-hidden="true">&raquo;</span>  
      <span class="sr-only">Next</span>  
    </a>  
  </li>  
</ul>  
{% endblock %}
```