esprit Se former autrement HONORIS UNITED UNIVERSITIES	**Année Universitaire 2024-2025**	Symfony

Workshop:

Génération de Données de Test avec Faker et Symfony 6

Objectifs :

- Comprendre comment utiliser Faker pour générer des données de test réalistes.
- Apprendre à créer et charger des fixtures dans Symfony.

Prérequis :

- Avoir un projet Symfony6 configuré.
- Avoir installé les dépendances nécessaires (fakerphp/faker, orm-fixtures, cocur/slugify).

Étapes :

- **Etape1 : Création de l'entité Student :**

Student
• id • Name

- **Etape2 : Modifier l'entité « Student » comme suit :**

1) Ajouter les deux champs «**updatedAt** » et «**createdAt** » :

```
    #[ORM\Column(type: 'datetime_immutable')]
        #[Assert\NotNull()]
        private \DateTimeImmutable $updatedAt;

        #[ORM\Column(type: 'datetime_immutable')]
        #[Assert\NotNull()]
        private \DateTimeImmutable $createdAt;
```

2) Ajouter le champ **« Slug »** : Slug unique pour l'étudiant, généré à partir du nom ainsi que les getters et les setters.

```
     #[ORM\Column(length: 255,
    unique: true)]
        #[Assert\NotNull()]
        #[Assert\NotBlank()]
```

```
        private ?string $slug =
    null;
    public function getSlug():
    ?string
        {
            return $this->slug;
        }

        public function
    setSlug(string $slug): static
        {
            $this->slug = $slug;
            return $this;
        }
```

L'ajout d'un champ slug dans votre entité Student présente plusieurs avantages, notamment en termes de référencement, de convivialité des URL et de gestion des données. Voici quelques raisons pour lesquelles vous pourriez vouloir ajouter un slug :

- **SEO (Search Engine Optimization)** : Les slugs sont souvent utilisés pour améliorer le référencement des pages web. Un slug bien structuré et descriptif peut aider les moteurs de recherche à comprendre le contenu de la page, ce qui peut améliorer le classement de votre site dans les résultats de recherche.
- Convivialité des URL : Les slugs rendent les URL plus lisibles et plus conviviales pour les utilisateurs. Par exemple, une URL comme example.com/student/john-doe est plus facile à comprendre et à mémoriser qu'une URL avec un identifiant numérique comme example.com/student/123.
- **Stabilité des URL :** Les slugs permettent de créer des URL stables qui ne changent pas même si d'autres attributs de l'entité changent. Cela est particulièrement utile pour les liens permanents (permalinks) dans les blogs et les sites de contenu.
- **Gestion des Données :** Les slugs peuvent être utilisés comme identifiants uniques pour les entités, ce qui facilite la recherche et la gestion des données. Par exemple, vous pouvez rechercher un étudiant par son slug plutôt que par son identifiant numérique.

- **Prévention des Conflits :**En utilisant des slugs uniques, vous évitez les conflits de noms dans votre base de données. Cela est particulièrement important lorsque vous avez des entités avec des noms similaires ou identiques.

3) Ajouter la méthode **prePersist :** c'est une méthode de cycle de vie d'une entité Doctrine. Elle est appelée automatiquement par Doctrine avant que l'entité ne soit persistée (enregistrée) dans la base de données pour la première fois.

```
    #[ORM\PrePersist]
        public function prePersist()
        {
            $this->createdAt = new \DateTimeImmutable();
            if ($this->name !== null) {
                $this->slug = (new Slugify())->slugify($this->name);
            } else {
                $this->slug = '';
            }
        }
```

4) Ajouter la méthode **preUpdate** : c'est une méthode de cycle de vie d'une entité Doctrine, similaire à **prePersist**, mais elle est appelée juste avant que l'entité ne soit mise à jour dans la base de données.

```
    #[ORM\PreUpdate]
        public function preUpdate()
        {
            $this->updatedAt = new \DateTimeImmutable();
        }

        public function getUpdatedAt(): \DateTimeImmutable
        {
            return $this->updatedAt;
        }
        public function setUpdatedAt(\DateTimeImmutable $updatedAt): self
        {
            $this->updatedAt = $updatedAt;
```

```
            return $this;
        }
```

- **Etape3: Mise en place des DataFixtures**

1) Pour pouvoir générer des données exemple pour la base de données, nous allons installer le composant orm-fixtures dans notre projet au moyen de la commande suivante :

```
composer require --dev orm-fixtures
```

⇨ Ce composant a créé un dossier **DataFixtures** dans src.

2) Afin de générer du contenu aléatoire, nous allons installer le composant fakerphp au moyen de la commande :

```
composer require fakerphp/faker
```

3) Création des Fixtures : Pour créer une Fixture, nous pouvons utiliser le maker bundle au moyen de la commande ci-dessous :

```
symfony console make:fixtures StudentsFixtures
```

⇨ Cette commande crée le fichier **src/DataFixtures/StudentsFixtures.php**

Nous allons le modifier pour créer une liste de 150 étudiants :

```
<?php
namespace App\DataFixtures;
use App\Entity\Student;
use Cocur\Slugify\Slugify;
use DateTimeImmutable;
use Faker\Factory;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;

class StudentsFixtures extends Fixture
{
    public function load(ObjectManager $manager): void
    {
```

```
            $faker = Factory::create('fr_FR');
            for ($i = 0; $i < 150; $i++) {
                $student = new Student();
                            $name = $faker->unique()->words(2, true); // Génère
    deux mots aléatoires
                 $student->setName($name);
                $student->setSlug((new Slugify())->slugify($name));
                $student->setCreatedAt( new DateTimeImmutable());
                $student->setUpdatedAt(new DateTimeImmutable());
                $manager->persist($student);
            }
            $manager->flush();

        }}
```

4) **Chargement des fixtures** : Pour charger les fixtures dans la base de données, nous allons utiliser la commande suivante :

```
Symfony console doctrine:fixtures:load
```

Références :

- DoctrineFixturesBundle Documentation