08

Consolidando o seu conhecimento

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula:

- Dentro da pasta **scr/Entity**, crie a classe `Curso`, com o `namespace Alura\Doctrine\Entity` e anotada com `@Entity`
  - Nesta classe, crie os atributos privados `id` (crie também seu *getter*), `nome` (crie também seu *getter* e *setter*) e `alunos` (crie também seu *getter*)
  - Anote o atributo `id` com `@Id`, `@GeneratedValue` e `@Column`, dizendo que a coluna será do tipo `integer`
  - Anote o atributo `nome` com `@Column`, dizendo que a coluna será do tipo `string`
  - Anote o atributo `alunos` com `@ManyToMany`, dizendo que a entidade alvo será `Aluno` e que a relação é invertida com `cursos`:

```
/**
 * @ManyToMany(targetEntity="Aluno", inversedBy="cursos")
 */
private $alunos;
```

- No construtor da classe `Curso`, instancie o atributo `alunos` como um `ArrayCollection`
- Além disso, implemente os métodos `addAluno`, que recebe um `Aluno` por parâmetro. Dentro do método, verifique se o `Aluno` recebido por parâmetro já está na coleção de alunos, e caso sim, nada será feito, então retorne a própria instância da classe. Mas se o `Aluno` não existir na coleção, adicione-o, atribua o seu `Curso` e retorne a própria instância da classe:

```
public function addAluno(Aluno $aluno)
{
    if ($this->alunos->contains($aluno)) {
        return $this;
    }

    $this->alunos->add($aluno);
    $aluno->addCurso($this);

    return $this;
}
```

- Na classe `Aluno`, adicione o atributo `cursos`. Anote-o com `@ManyToMany`, configurando que a entidade alvo será `Curso` e qual propriedade de `Curso` se refere à classe:

```
/**
* @ManyToMany(targetEntity="Curso", mappedBy="alunos")
*/
private $cursos;
```

  - No construtor da classe, instancie o atributo `cursos` como um `ArrayCollection`
  - Implemente os métodos `getCursos`, que retorna a coleção de cursos, e `addCurso`, que recebe um `Curso` por parâmetro. Dentro do método, verifique se o `Curso` recebido por parâmetro já está na coleção de cursos, e caso

sim, nada será feito, então retorne a própria instância da classe. Mas se o `Curso` não existir na coleção, adicione-o, atribua o seu `Aluno` e retorne a própria instância da classe:

```
public function addCurso(Curso $curso): self
{
    if ($this->cursos->contains($curso)) {
        return $this;
    }

    $this->cursos->add($curso);
    $curso->addAluno($this);

    return $this;
}

public function getCursos(): Collection
{
    return $this->cursos;
}
```

- Dentro da pasta **commands**, crie o arquivo **criar-curso.php**. Neste arquivo, receba, por parâmetro na linha de comando, o nome do curso e implemente o código para inseri-lo no banco de dados:

```
<?php

use Alura\Doctrine\Entity\Curso;
use Alura\Doctrine\Helper\EntityManagerFactory;

require_once __DIR__ . '/../vendor/autoload.php';

$entityManagerFactory = new EntityManagerFactory();
$entityManager = $entityManagerFactory->getEntityManager();

$curso = new Curso();
$curso->setNome($argv[1]);

$entityManager->persist($curso);
$entityManager->flush();
```

- Novamente dentro da pasta **commands**, crie o arquivo **vincular-aluno-curso.php**. Neste arquivo, receba, por parâmetro na linha de comando, o id do aluno e o id do curso e busque-os, através do `EntityManager`. Em seguida, adicione o aluno ao curso e envie as alterações para o banco de dados:

```
<?php

use Alura\Doctrine\Entity\Aluno;
use Alura\Doctrine\Entity\Curso;
use Alura\Doctrine\Helper\EntityManagerFactory;

require_once __DIR__ . '/../vendor/autoload.php';

$entityManagerFactory = new EntityManagerFactory();
$entityManager = $entityManagerFactory->getEntityManager();
```

```
$idAluno = $argv[1];
$idCurso = $argv[2];

/** @var Curso $curso */
$curso = $entityManager->find(Curso::class, $idCurso);
/** @var Aluno $aluno */
$aluno = $entityManager->find(Aluno::class, $idAluno);

$curso->addAluno($aluno);

$entityManager->flush();
```

- Na linha de comando, gere uma *migration*, comparando o seu banco de dados atual com as informações de mapeamento:

```
vendor\bin\doctrine-migrations migrations:diff
```

- Em seguida, execute as *migrations*:

```
vendor\bin\doctrine-migrations migrations:migrate
```

- Por último, crie alguns cursos e vincule-os aos alunos existentes no seu banco de dados

Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.