10

Consolidando o seu conhecimento

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Com o PHP e Composer instalados, acesse pela linha de comando a pasta do projeto baixado (caso você ainda não tenha feito o download, faça-o aqui (https://caelum-online-public.s3.amazonaws.com/1483-symfony-parte-3/01/projeto-inicial.zip)) e digite: `composer install` .

2) No arquivo `.env` , altere o parâmetro `DATABASE_URL` para que reflita as configurações do seu banco de dados.

3) Habilite a extensão `pdo_mysql` em sua instalação do PHP.

4) Novamente na linha de comando, digite: `php bin/console doctrine:database:create` e depois `php bin/console doctrine:migrations:migrate` .

5) Digite agora `php -S localhost:8001 -t public` para subir o servidor web do PHP (a porta 8001 pode ser alterada livremente).

6) Abra o Postman e tente acessar a URL `localhost:8001/qualquer_coisa` . Note que há um erro exibido em HTML.

7) Crie um arquivo chamado `ExceptionHandler.php` na pasta `src/EventListeners` , com o seguinte conteúdo:

```
<?php

namespace App\EventListener;

use App\Entity\HypermidiaResponse;
use App\Helper\EntityFactoryException;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;

use Symfony\Component\HttpKernel\KernelEvents;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpKernel\Event\ExceptionEvent;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

class ExceptionHandler implements EventSubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return [
            KernelEvents::EXCEPTION => [
                ['handleEntityFactoryException', 1],
                ['handle404Exception', 0],
            ],
        ];
    }

    public function handle404Exception(ExceptionEvent $event)
    {
        if ($event->getException() instanceof NotFoundHttpException) {
            $response = HypermidiaResponse::fromError($event->getException())->getResponse();
            $response->setStatusCode(Response::HTTP_NOT_FOUND);
```

```
                $event->setResponse($response);
            }
        }


        public function handleEntityFactoryException(ExceptionEvent $event)
        {
            if ($event->getException() instanceof EntityFactoryException) {
                $response = HypermidiaResponse::fromError($event->getException())->getResponse();
                $response->setStatusCode(Response::HTTP_BAD_REQUEST);
                $event->setResponse($response);
            }
        }
    }
```

8) Acesse a mesma URL e veja que o erro agora é exibido em JSON.

9) Em um outro terminal, digite o seguinte comando: `php bin/console doctrine:fixtures:load` .

10) Acesse a URL `localhost:8001/login` , utilizando o verbo POST e passando (em JSON) os valores de `username` e `password` . Copie o `access_token` que é devolvido.

11) Tente criar uma especialidade, escrevendo `descricao` no JSON de forma errada (`descsricao` , por exemplo). Acesse `localhost:8001/especialidades` , utilizando o verbo POST sem passar a descrição. Observe que um erro em HTML é exibido.

12) Na pasta `src/Helper` , crie o arquivo `EntityFactoryException.php` , com o seguinte conteúdo:

```
<?php

namespace App\Helper;

class EntityFactoryException extends \Exception { }
```

13) Altere a classe `EspecialidadeFactory` para que o seu conteúdo seja o seguinte:

```
<?php

namespace App\Helper;

use App\Entity\Especialidade;

class EspecialidadeFactory implements EntityFactoryInterface
{
    public function createEntity(string $json): Especialidade
    {
        $objetoJson = json_decode($json);
        $this->checkIfDescriptionExists($objetoJson);

        $especialidade = new Especialidade();
        $especialidade->setDescricao($objetoJson->descricao);

        return $especialidade;
    }
```

```
    /**
     * @param $objetoJson
     * @throws EntityFactoryException
     */
    private function checkIfDescriptionExists($objetoJson): void
    {
        if (!property_exists($objetoJson, 'descricao')) {
            throw new EntityFactoryException('A descrição de uma especialidade é obrigatória');
        }
    }
}
```

14) Tente criar a especialidade com `descricao` estando escrito de forma errada novamente. Veja que o erro agora está em JSON.

15) Faça o mesmo na classe `MedicoFactory`, tendo o seguinte resultado:

```
<?php

namespace App\Helper;

use App\Entity\Medico;
use App\Repository\EspecialidadeRepository;

class MedicoFactory implements EntityFactoryInterface
{
    /**
     * @var EspecialidadeRepository
     */
    private $especialidadeRepository;

    public function __construct(EspecialidadeRepository $especialidadeRepository)
    {
        $this->especialidadeRepository = $especialidadeRepository;
    }

    public function createEntity(string $json): Medico
    {
        $objetoJson = json_decode($json);
        $this->checkIfAllPropertiesExist($objetoJson);

        $especialidade = $this->especialidadeRepository->find($objetoJson->especialidadeId);
        if (is_null($especialidade)) {
            throw new EntityFactoryException('Especialidade inexistente');
        }

        $medico = new Medico();
        $medico
            ->setNome($objetoJson->nome)
            ->setCrm($objetoJson->crm)
            ->setEspecialidade($especialidade);

        return $medico;
    }

    private function checkIfAllPropertiesExist(object $objetoJson): void
```

```
        {
            if (!property_exists($objetoJson, 'nome')) {
                throw new EntityFactoryException('Médico precisa de nome');
            }

            if (!property_exists($objetoJson, 'crm')) {
                throw new EntityFactoryException('Médico precisa de CRM');
            }

            if (!property_exists($objetoJson, 'especialidadeId')) {
                throw new EntityFactoryException('Médico precisa de especialidade');
            }
        }
    }
```