Gérer les exceptions dans Laravel 9:

1.Introduction :

Lorsque vous démarrez un nouveau projet Laravel, la gestion des erreurs et des exceptions est déjà configurée pour vous. La classe **App\Exceptions\Handler** est l'endroit où toutes les exceptions lancées par votre application sont enregistrées et ensuite présentées à l'utilisateur.

2.Configuration

L'option de débogage de votre fichier de configuration **config/app.php** détermine la quantité d'informations relatives à une erreur qui sont effectivement affichées à l'utilisateur. Par défaut, cette option est définie pour respecter la valeur de la variable d'environnement **APP_DEBUG**, qui est stockée dans votre fichier **.env**.

Pendant le développement local, vous devez définir la variable d'environnement **APP_DEBUG** sur **true**. Dans votre environnement de production, cette valeur doit toujours être **false**. Si la valeur est définie sur **true** en production, vous risquez d'exposer des valeurs de configuration sensibles aux utilisateurs finaux de votre application.

3.Le gestionnaire d'exceptions

Il vous est possible de créer vos propres exceptions dans Laravel grâce à la commande suivante :

php artisan make :exception StudentNotFoundException

3.1 Exceptions à signaler et à rendre

Vous pouvez définir des méthodes ***report*** et ***render*** directement sur vos exceptions personnalisées. Lorsque ces méthodes existent, elles sont automatiquement appelées par le framework :

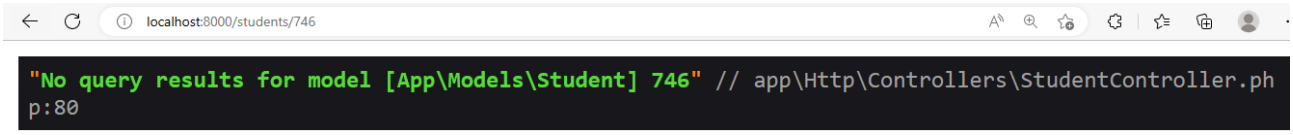
```php
<?php

namespace App\Exceptions;

use Exception;

class StudentNotFoundException extends Exception
{
    public function report(){
        return false;

    }
    public function render(){
   return view('errors.studentnotfound');


    }

    public function context()
    {
        return ['order_id' =>45];
    }

}
```

Cette class hérite d'une autre Class Exception. On peut également remarquer qu'elle contient 2 méthodes :
report() : permet de faire des logs de l'exception ainsi que de les envoyer vers des services extérieurs (par exemple par mail).
render() : permet de faire un rendu de l'exception en réponse HTTP.

Dans notre code nous allons maintenant faire appel à cette Exception :

```
public function show(Request $request,$student)
  {
    try {
    $student=Student::findOrFail($student);
    } catch (ModelNotFoundException $e) {
            // dd($e->getMessage());

       throw new studentNotFoundException();
    }
     return view('students.profile', ['student' => $student]);
```

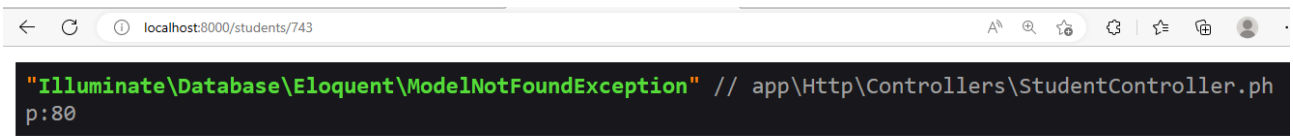
//Output



Après avoir mis en place le bloc **try** et **catch** il nous faut maintenant lancer l'exception avec **throw** qui nous permettra de créer une instance d'Exception. On y passe en paramètre le message que l'on souhaite retourner.

Exemple : Afficher la classe de l'exception déclenchée

```
public function show(Request $request,$student)
  {
    try {
    $student=Student::findOrFail($student);
    } catch (ModelNotFoundException $e) {
       dd(get_class($e));
    }

     return view('students.profile', ['student' => $student]);
  }
```



3.2 Contexte global de journalisation

S'il est disponible, Laravel ajoute automatiquement l'ID de l'utilisateur actuel au message de journal de chaque exception en tant que donnée contextuelle. Vous pouvez définir vos propres données contextuelles globales en surchargeant la méthode contextuelle de la classe **App\Exceptions\Handler** de votre application. Ces informations seront incluses dans chaque message de journal d'exception écrit par votre application :

App\Exceptions\Handler.php

```
protected function context()
{
    return ['foo' => 'bar'];
}
```

3.3 Contexte du journal des exceptions

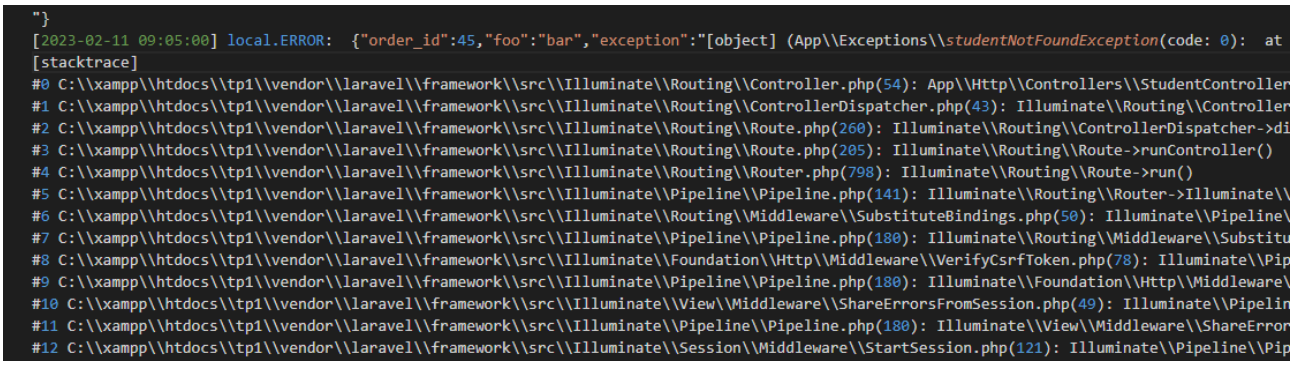
Si l'ajout d'un contexte à chaque message de journal peut être utile, il arrive qu'une exception particulière présente un contexte unique que vous souhaitez inclure dans vos journaux. En définissant une méthode de contexte sur l'une des exceptions personnalisées de votre application, vous pouvez spécifier toute donnée pertinente pour cette exception qui doit être ajoutée à l'entrée du journal de l'exception :

App\Exceptions\StudentNotFoundException.php

```
<?php

namespace App\Exceptions;

use Exception;

class StudentNotFoundException extends Exception
{

    public function context()
    {
```

```
        return ['order_id' => 45];
    }
}
```

Voici le journal des exceptions (**\storage\logs\laravel.log**)



4.Pages d'erreur HTTP personnalisées :

Vous voulez personnaliser la page d'erreur pour le code d'état HTTP 404, alors créez un fichier 404.blade.php dans le répertoire **resources/views/errors**. Laravel exécutera cette page sur toutes les erreurs 404 générées par votre application. Vous pouvez également créer une vue nommée le code d'état HTTP. Il passera la fonction d'interruption à la vue comme une variable d'exception.

```
{{ $exception->getMessage() }}
```