本ステップでおこなうこと

例外(Exception)・エラー(Error)が起こったときの共通処理を実装します。

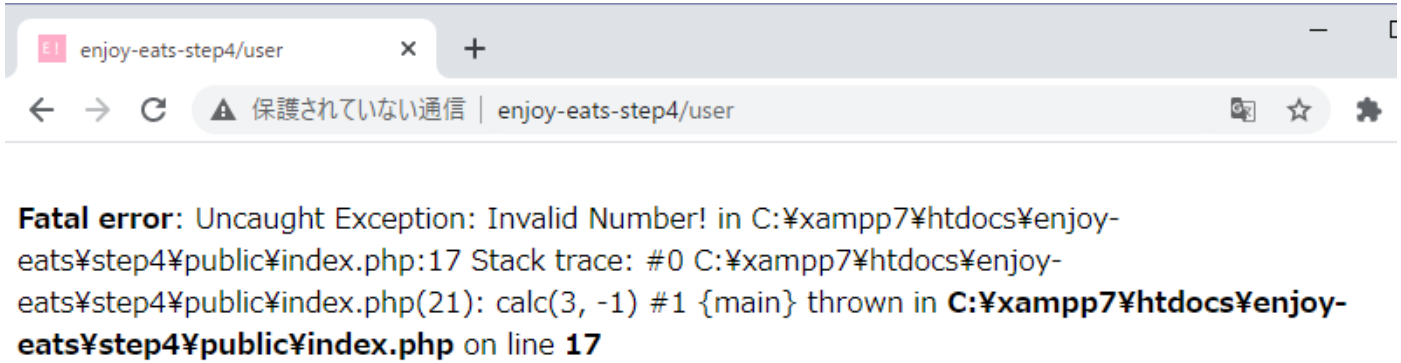
catchされなかった例外/エラーの扱い(1)

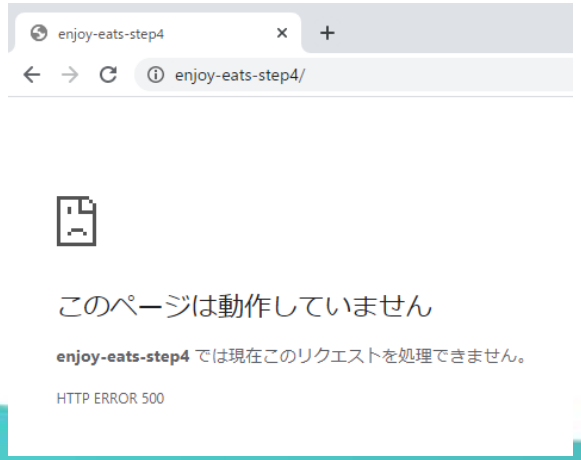
このコードを実行すると…

```
function calc(int $a, int $b)
{
    if ($a <= 0 || $b <= 0) {
        throw new Exception('Invalid Number!');
    }
    return $a + $b;
}
calc(3, -1);
```

catchされなかった例外/エラーの扱い(2)

Exceptionがcatchされないために、Fatal errorが発生し、
エラーメッセージがそのまま表示される





← display_errors=offのときの表示

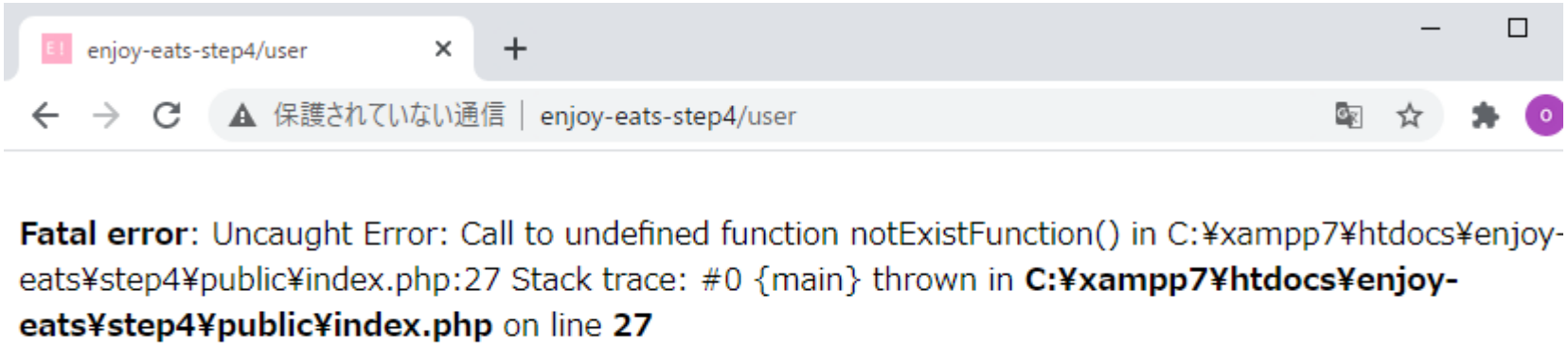
catchされなかった例外/エラーの扱い(3)

このコードを実行すると…

```
// 存在しない関数を呼び出す。
$result = notExistFunction();
```

catchされなかった例外/エラーの扱い(4)

やはり、Fatal Errorとなり、エラーメッセージがそのまま表示される



catchされなかった例外/エラーを統一的に扱う

そこで登場するのが、catchされなかった例外/エラーを統一的に扱うための、set_exception_handler関数です。

set_exception_handler (callable $handler)

set_ exception_handler()を使うことで生まれるメリット

メリット1	きちんとデザインされたエラーページを表示することができる。お問い合わせ先なども入れることができる。
メリット2	例外/エラーの詳細を、サイト運営者のためにログ出力したり、メール通知することができる。
メリット3	例外/エラーの詳細を、ユーザの目から隠すことができる。

本ステップのクラス構成

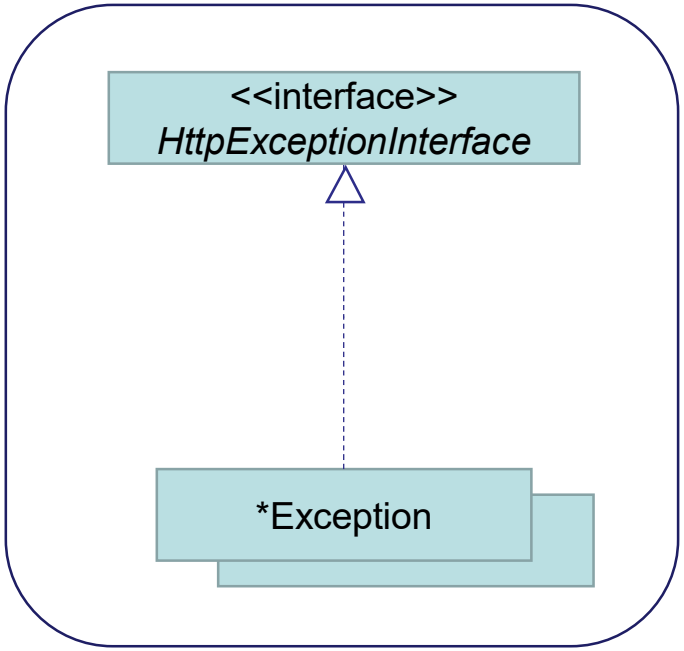
- set_exception_handler関数にエラー処理を登録しておく。
- ルーティング情報に存在しないURLにアクセスされたときに、HttpExceptionInterface例外を投げるように実装を変える。

index.php

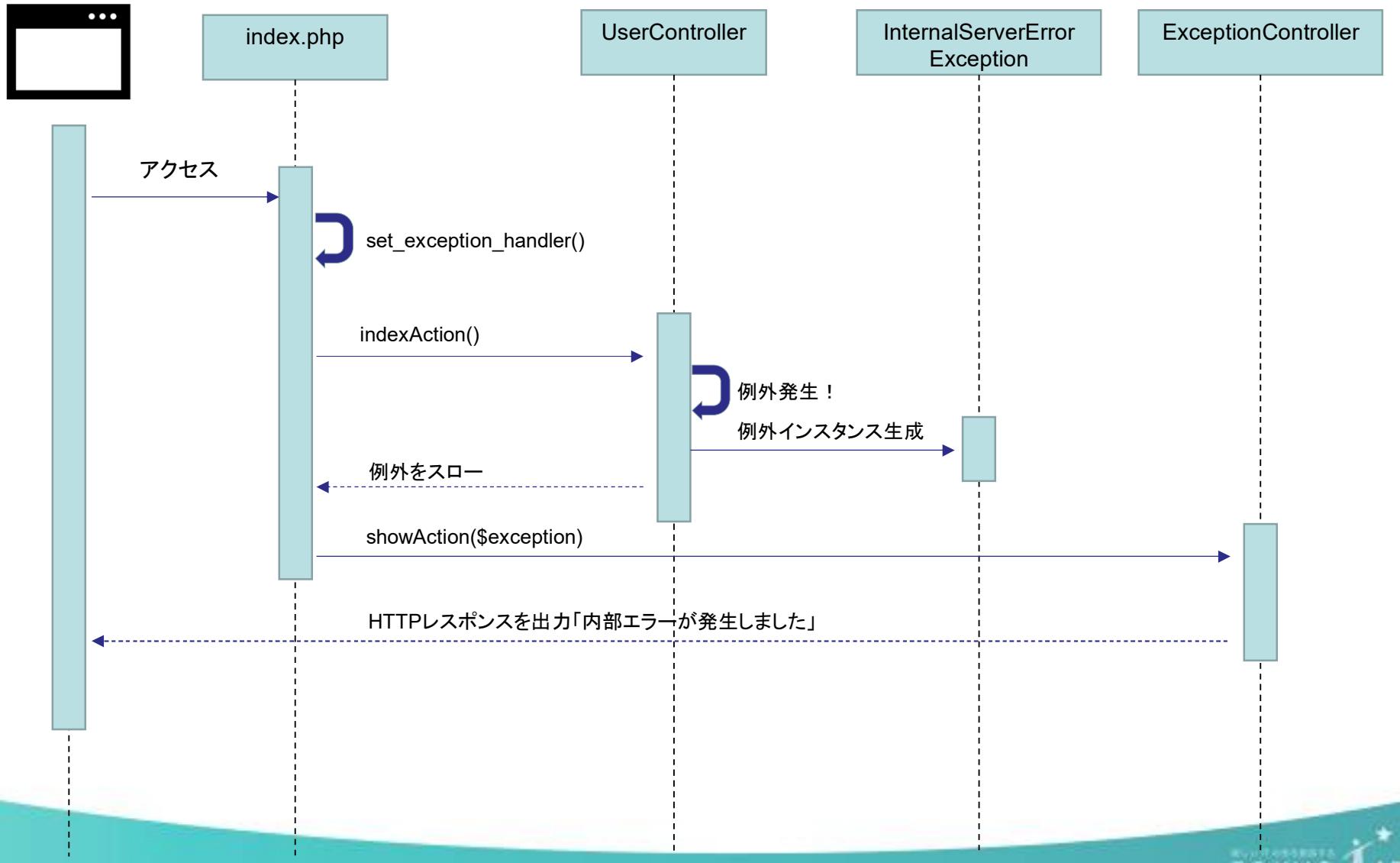
App¥Modules¥Common¥Controllers¥
ExceptionController

エラーページ表示用の
コントローラークラス

例外クラス群
(App¥Libs¥Core¥Exception)

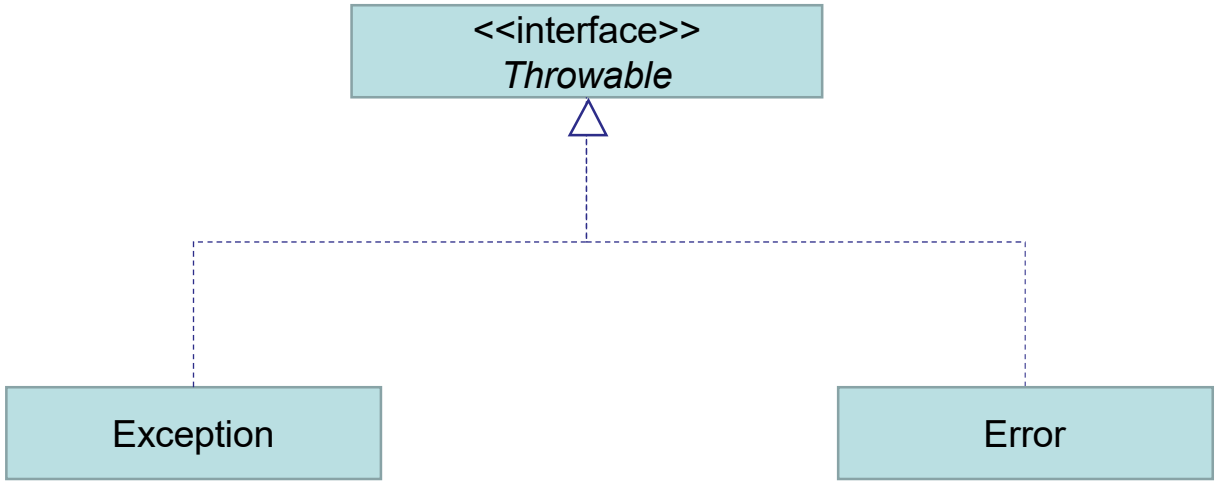


本ステップの処理の流れ
(一例として、UserControllerで例外発生時)



本ステップの補足(1)

PHP7以降は、大半のエラーはErrorクラスとして扱われており、
set_exception_handlerはErrorクラスも検知することができます。



本ステップの補足(2)

ただし、一部の致命的なエラーは、Errorクラスをスローしない、PHP5以前のままの実装で残っています。このエラーも検知するときは、set_error_handlerを使います。

```
$handleError = function (int $errorNo, string $errorMessage) {
    $e = new ¥Exception($errorMessage);
    (new App¥Libs¥Core¥ExceptionController())->showAction($e);
};
set_error_handler($handleError);
```

メモリ不測やタイムアウトのときは、そもそもset_error_handlerの命令行まで到達しないこともあることに注意してください。

```
trigger_error('エラー発生', E_USER_ERROR);
```

とすることで、強引にエラーを発生させることもできます。

本ステップの変更ファイル一覧

●追加したファイル

- app/Modules/Common/Controllers/ExceptionController.php
  → エラーページ用のコントローラークラス

- app/Libs/Core/Exception/*Exception.php

●変更したファイル

- public/index.php
  → キャッチされなかった例外があったときにExceptionControllerを呼び出す処理を追加

参考情報

- PHP本格入門(上)
  「3-7 イレギュラーなケースに対処する - 例外処理」