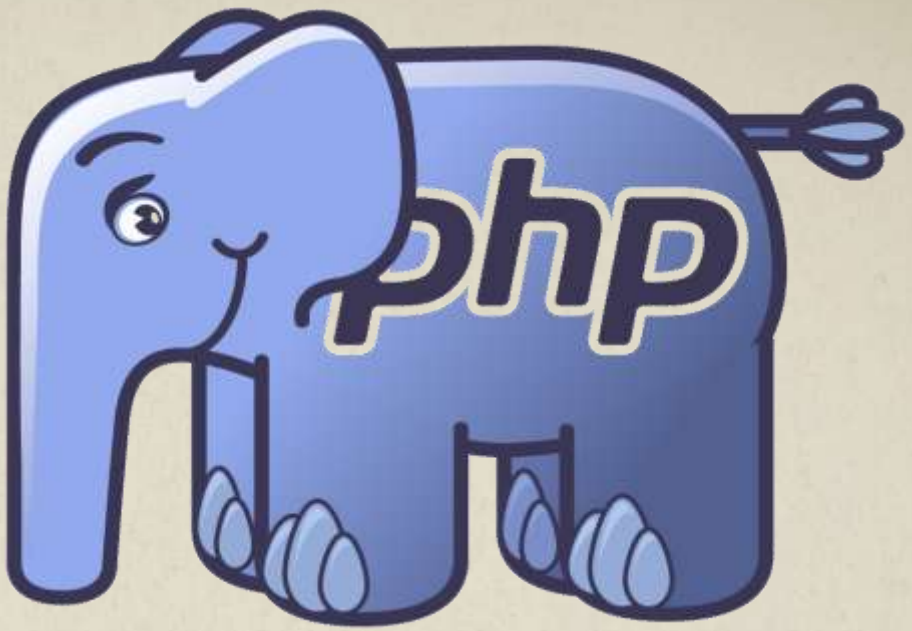


PHP: ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

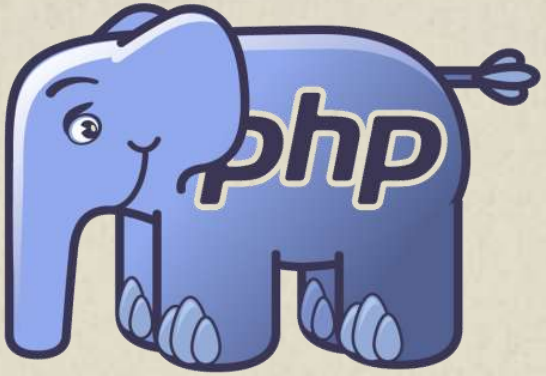
УРОК 5. ИСКЛЮЧЕНИЯ

ОБРАЩЕНИЕ К «СКЛАДЧИКАМ»

Я считаю складчины – полной ерундой. Невозможно научиться чему-то, просматривая видео. Без домашних заданий, без общения с преподавателями и коллегами. Покупая в складчину видеозаписи курсов, вы вредите прежде всего самим себе, создавая иллюзию «обучения». И поддерживаете каких-то мутных личностей-«организаторов», имеющих свой процент.

Впрочем, дело ваше.

Однако, если вы хотите по-настоящему учиться – приходите. Адрес есть на слайдах. Напишите в поддержку, мол «я складчик, но я хочу учиться». Скидку гарантирую ☺



ИСКЛЮЧЕНИЯ

ПОНЯТИЕ ИСКЛЮЧЕНИЯ

Обработка исключительных ситуаций — механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма.

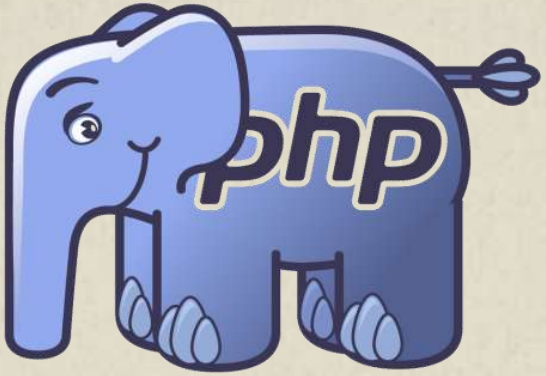
- **ИСКЛЮЧЕНИЕ** – это ситуация, при которой дальнейшее нормальное выполнение невозможно или бессмысленно.
- **ИСКЛЮЧЕНИЕ**, в отличие от, например, фатальной ошибки, ожидаемо нами. Мы готовы к его появлению и к обработке исключительной ситуации.
- **ИСКЛЮЧЕНИЕ** – это всегда ситуация, возникающая во время исполнения программы. Ошибки парсера или компилятора никак не могут называться исключениями, поскольку к этому моменту программа еще не запущена.

ИСКЛЮЧЕНИЯ

ПРИМЕРЫ ИСКЛЮЧЕНИЙ

- **ДЕЛЕНИЕ НА НОЛЬ** – ни дальнейшие вычисления, ни попытка использовать результат операции не приведут ни к чему хорошему
- **РАЗРЫВ СОЕДИНЕНИЯ С БД** – исключительная ситуация, которая может возникнуть в любой момент и сделать бессмысленной или даже вредной дальнейшую нормальную работу программы.
- **ОТСУТСТВИЕ нужного файла,** из которого предполагалось чтение
- **НЕВЕРНЫЙ пароль** в форме входа
- **БЕССМЫСЛИЦА в поле для поиска по сайту**
- **ОТСУТСТВИЕ пользователя, которому предназначается сообщение**

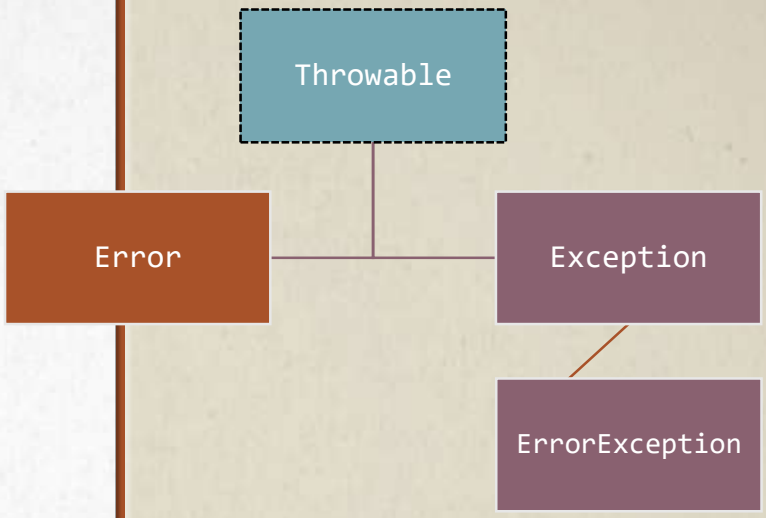
ИСКЛЮЧЕНИЯ



ИСКЛЮЧЕНИЯ В PHP

Существуют два базовых класса: `Error` и `Exception`

- Первый класс использует сам PHP. Не напрямую, а наследуя от него ряд других классов:
  - **`ArithmeticError`**
  - **`AssertionError`**
  - **`DivisionByZeroError`**
  - **`ParseError`**
  - **`TypeError`**
- Второй класс является базовым для всех пользовательских исключений. Вы можете наследовать свои классы от него.
- У этих двух классов есть нечто общее. Это интерфейс **`Throwable`**.
  Его нельзя реализовать напрямик!



КЛАССЫ ИСКЛЮЧЕНИЙ И ИЕРАРХИЯ

Раз есть класс, значит есть и объекты!

- Объект исключения можно создать с помощью обычного конструктора соответствующего класса исключения:

```
$ex1 = new Exception;
$ex2 = new Exception('DB error');
$ex3 = new Exception('DB error',
42);
```

- Объект-исключения это вполне себе обычный объект. Его можно присвоить переменной, передать как аргумент, вызывать его методы:

```
echo $ex2->getMessage();
$code = $ex3->getCode();
```

и даже

```
echo $ex1;
```

ОБЪЕКТ-ИСКЛЮЧЕНИЕ

И эти объекты можно бросить!

```
throw new Exception('Aaaa!!!');
```

ВЫБРОС ИСКЛЮЧЕНИЯ

И самое главное – поймать!

```
try {

   // некий код, который может
   // выбросить исключение

} catch (Exception $ex) {

   // обработка пойманного
   //  исключения

}
```

- Нужно обязательно указать класс «ловимого» исключения
- Будут пойманы те исключения, которые принадлежат этому классу или его наследникам
- Самое интересное – в catch опять можно бросить исключение! И оно будет всплывать выше…

КАК ПОЙМАТЬ ИСКЛЮЧЕНИЕ?

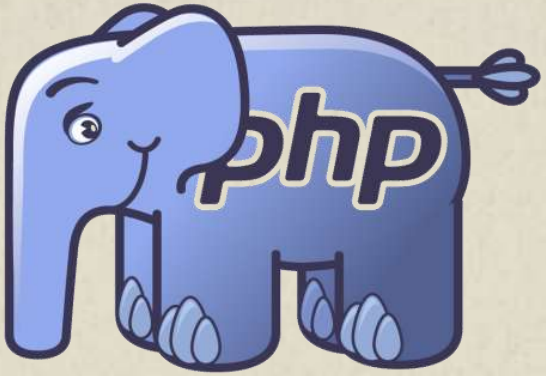
Несколько catch и finally

Есть механизм, который позволяет выполнить разный код для разных классов исключений

```
try {
  ...
} catch (Exception1 $ex) {
  // some code with Exception1
} catch (Exception2 $ex) {
  // some code with Exception1
}
```

Или выполнить код после обработки всех исключений (или даже если их не было)

```
try {
  ...
} catch (Exception $ex) {
 ...
} finally {
  // ффух, закончили!
}
```



ПРАКТИКА ПРИМЕНЕНИЯ

Где в PHP есть исключения?

- Во-первых в SPL (Standard PHP Library). Она содержит в себе десяток готовых стандартных классов исключений:
  - `BadMethodCallException`
  - `InvalidArgumentException`
  - `RangeException`
  - `InvalidArgumentException`
  - и другие

- Во-вторых, например, в PDO!

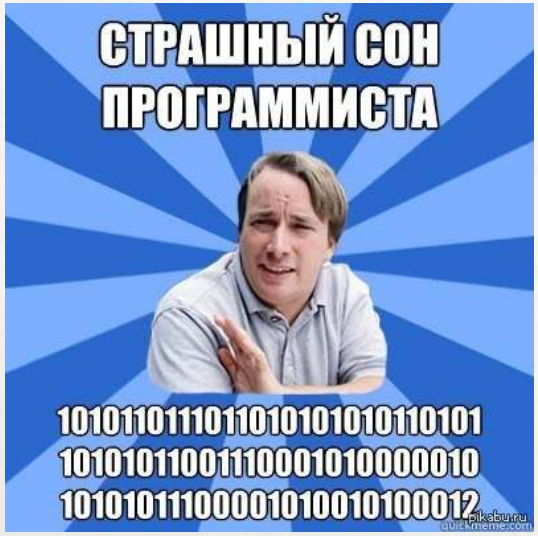
```
try {

  $dbh = new PDO($dsn);

} catch (PDOException $e) {
  echo 'Ошибка БД:' .
    $e->getMessage();
}
```

ИСКЛЮЧЕНИЯ В СТАНДАРТНОЙ БИБЛИОТЕКЕ

ИСКЛЮЧЕНИЯ В СТАРЫХ ВЕРСИЯХ PHP

В версиях до PHP 7 есть ряд особенностей, о которых надо знать!

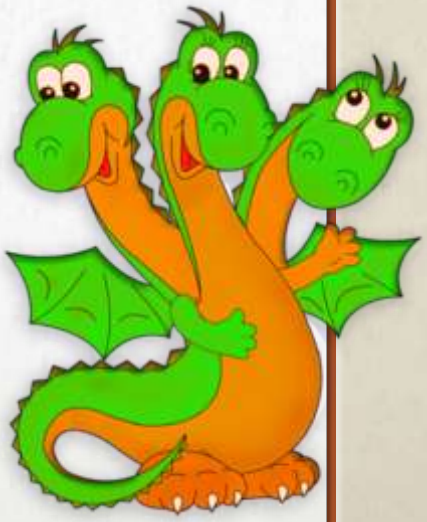
- Нет исключения **`Error`** и его наследников.
- Нет общего интерфейса **`Throwable`**
- Исключения ваших классов начинают наследование от **`Exception`**
- Многие ошибки самого PHP являются ошибками, а не исключениями



Мультиисключение: забавное архитектурное упражнение

- Это исключение
- Которое в свою очередь является коллекцией других исключений
- Позволяет добавлять исключения в себя, получать список, удалять

Весьма неплохо применяется для валидации форм, моделей etc, в тех случаях, когда нужно хранить информацию о **НЕСКОЛЬКИХ** исключительных ситуациях сразу



МУЛЬТИ ИСКЛЮЧЕНИЯ

Как всё-таки применять исключения?

- Во-первых определите свою иерархию исключений. Сделайте исключения моделей, например. Или исключение «не найдена запись в таблице». Или исключения для введенных пользователем данных.
- Определите «опасный код». Убедитесь, что вы выбрасываете исключения во всех исключительных ситуациях.
- Ловите все исключения. Старайтесь ловить их раньше, не давая всплывать до верхних слоев.
- Убедитесь, что вы ловите ВСЕ исключения! (в этом правиле не должно быть исключений ☺)

КАК РАБОТАТЬ С ИСКЛЮЧЕНИЯМИ ?

ДО ВСТРЕЧИ НА СЛЕДУЮЩЕМ УРОКЕ!

**ВИДЕОЗАПИСЬ, СЛАЙДЫ, ПРЕЗЕНТАЦИЯ
И ДОМАШНЕЕ ЗАДАНИЕ
БУДУТ ВЫЛОЖЕНЫ ДО 10 УТРА СЛЕДУЮЩЕГО ДНЯ**

ProfIT