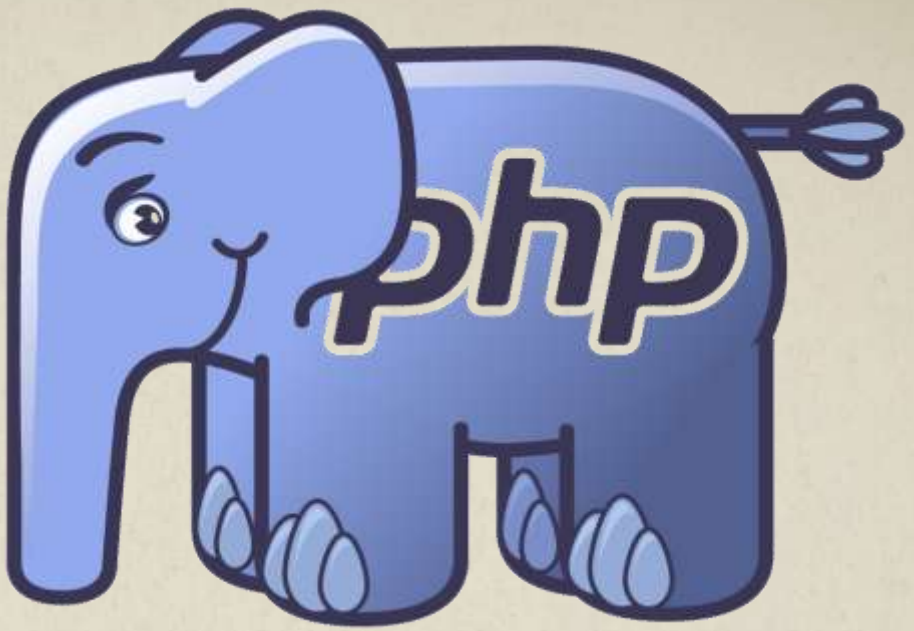


PHP: ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

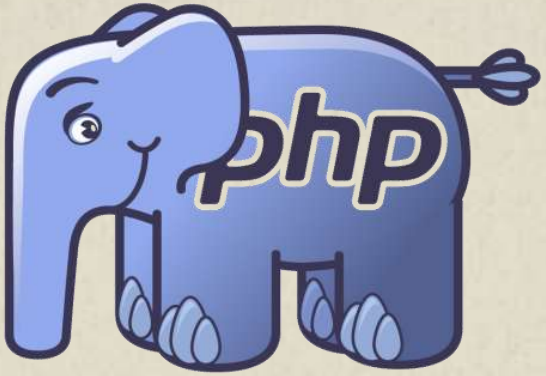
УРОК 2. ПРОДОЛЖЕНИЕ РАЗГОВОРА О МОДЕЛЯХ

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 ☺



ИНТЕРФЕЙСЫ

Интерфейс - это специальная сущность языка PHP, своего рода «контракт», который обязан выполнить класс.

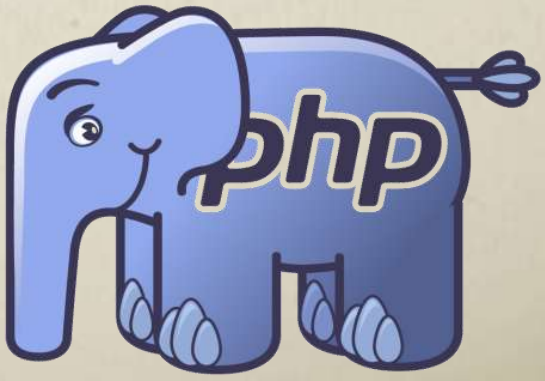
Интерфейс содержит в себе описание **публичных методов**, (*и констант*) которые ОБЯЗАН иметь класс, реализующий этот интерфейс:

```
interface Orderable
{
  public function getPrice();
  public function getWeight();
}
```

```
class Item
  implements Orderable
{
  public function getPrice() {
    ...
  }
  public function getWeight() {
    ...
  }
}
```

NB. *С точки зрения автозагрузки интерфейс – это класс. Полное имя интерфейса будет передано в функцию автозагрузки.*

ИНТЕРФЕЙСЫ



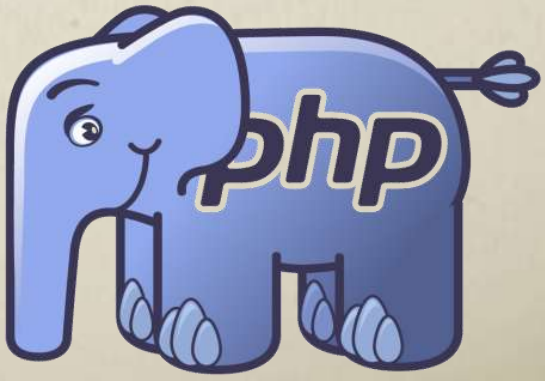
Интерфейсы могут, как классы, наследоваться друг от друга. Впрочем, это имеет ограниченное применение:

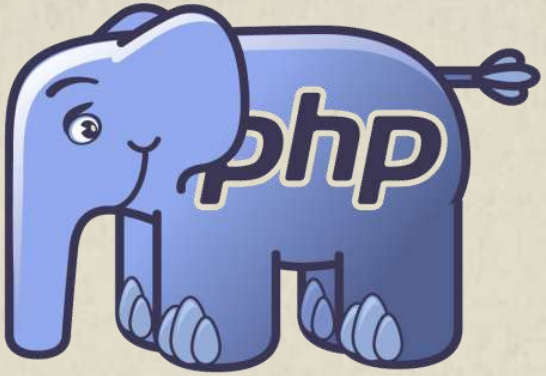
```
interface HasPrice
{
  public function getPrice();
}

...

interface Orderable
  extends HasPrice, HasWeight
{
}

class Item
  implements Orderable ...
```

Интереснее другое – класс может реализовывать несколько интерфейсов!

```
class Item
  implements HasPrice, HasWeight {

  public function getPrice() { ... }

  public function getWeight() { ... }

}
```

ИНТЕРФЕЙСЫ





ТАЙП-ХИНТИНГ

Тайп-хинтинг - это возможность указать ожидаемый ТИП аргумента функции.

1. **Скалярный**

Используем названия типов `bool`, `int`, `float`, `string`

```
function sum(int $a, int $b) {
  return $a + $b;
}

echo sum(1.2, 2.3); // 3
```

2. **Массив**

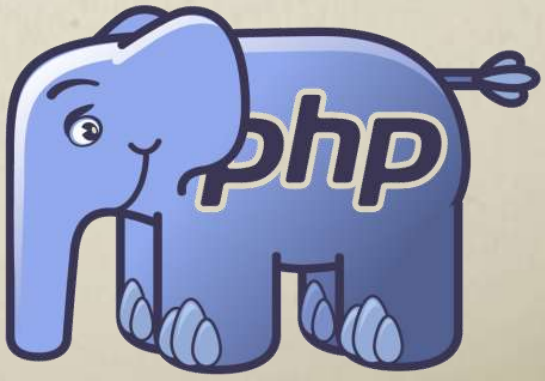
Используем тип `array`

```
function sum(array $numbers=[]) {
  ...
}
```

Возможен «строгий режим» контроля типов (TypeError вместо приведения):

```
declare(strict_types=1);

echo sum(1.2, 2.3); // Fatal error!
```

ТАЙП-ХИНТИНГ



Тайп-хинтинг - это возможность указать ожидаемый КЛАСС аргумента функции.

1. Точное соответствие имени класса классу объекта

```
function send(User $u, $message) {
  return $a + $b;
}
```

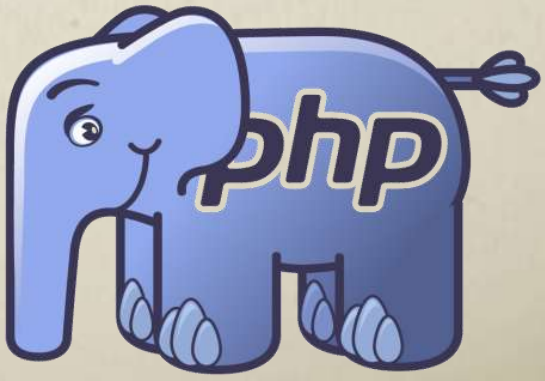
2. Класс-родитель

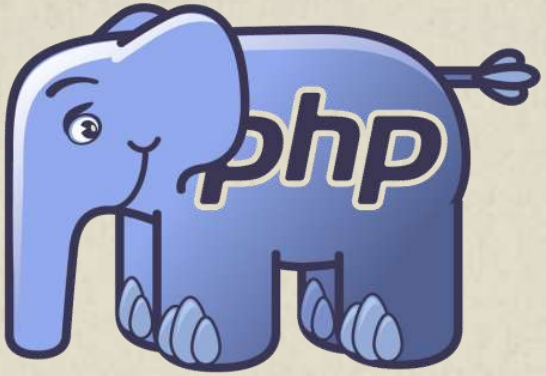
```
function send(User $u, $message) {
  ...
}

class Admin extends User { ...}

send (new Admin, 'Hello!');
```

3. И, наконец, интерфейс!!!

```
function send(HasEmail $u, $message) {
  ...
}

send($user, 'Hello!');
```

ТАЙП-ХИНТИНГ



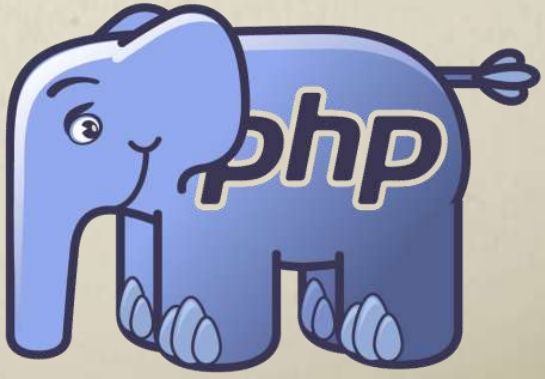


ПАТТЕРН «ОДИНОЧКА»

Singleton - это паттерн-трюк

Как написать класс, чтобы было возможным создание ТОЛЬКО ОДНОГО объекта данного класса?

SINGLETON

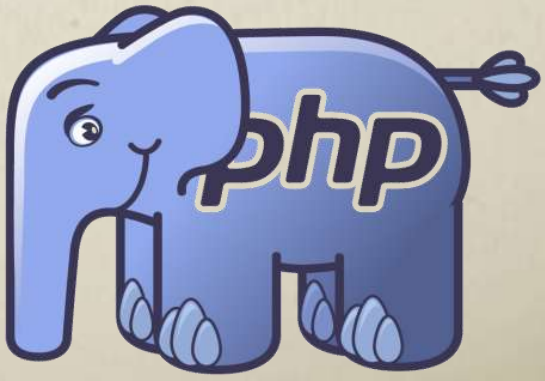


Singleton - это паттерн-трюк

Как написать класс, чтобы было возможным создание ТОЛЬКО ОДНОГО объекта данного класса?

1. Запретить нормальное создание объектов этого класса. Например: сделав конструктор непубличным.

SINGLETON

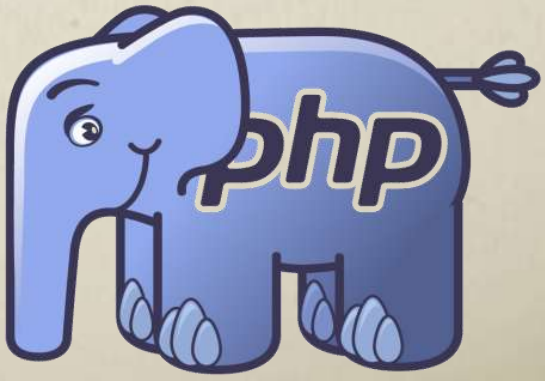


Singleton - это паттерн-трюк

Как написать класс, чтобы было возможным создание ТОЛЬКО ОДНОГО объекта данного класса?

1. Запретить нормальное создание объектов этого класса. Например: сделав конструктор непубличным.
2. Предусмотреть статический метод в классе, который будет возвращать объект.

SINGLETON



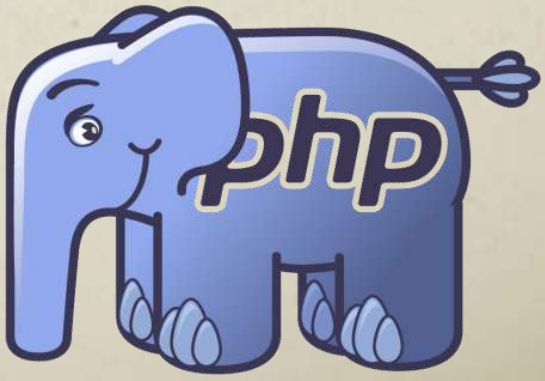
Singleton - это паттерн-трюк

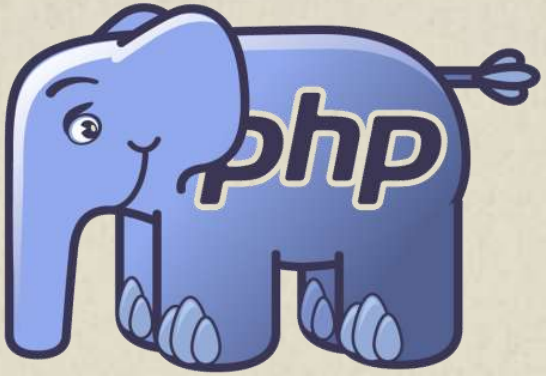
Как написать класс, чтобы было возможным создание ТОЛЬКО ОДНОГО объекта данного класса?

1. Запретить нормальное создание объектов этого класса. Например: сделав конструктор непубличным.
2. Предусмотреть статический метод в классе, который будет возвращать объект.
3. Поручить ему «считать» число объектов:
    1. Возвращать уже существующий, если есть.
    2. Или новый, если еще не было.

ProfIT!!!

SINGLETON





ТРЕЙТЫ

Трейт - это специальная сущность языка PHP, своего рода «заготовка», которую можно вставить в класс

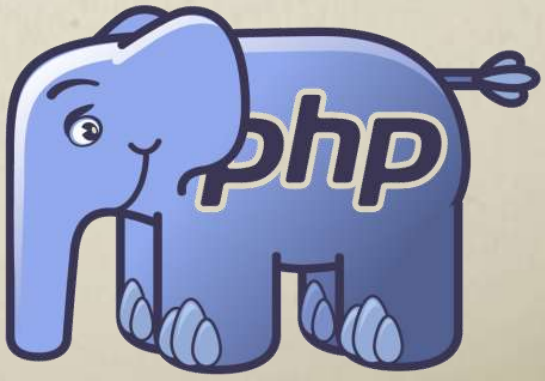
Трейт может содержать в себе методы и свойства, динамические и статические.

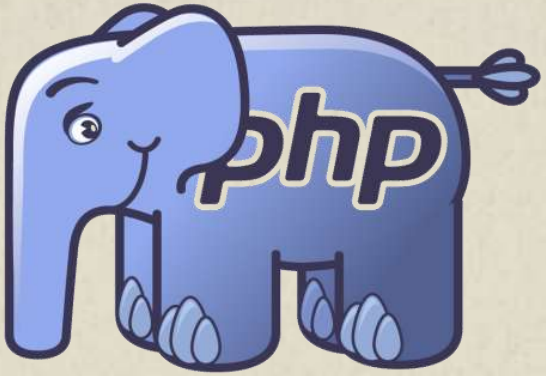
При компиляции вашей программы текст трейта будет вставлен в текст класса (ЭТО УПРОЩЕНИЕ, НО БЛИЗКОЕ!)

```
trait DateTime
{
  protected $started;
  protected $finished;
  public function getDuration() {
    return $finished - $started;
  }
}

class Order {
  use DateTime;
}
```

NB. *С точки зрения автозагрузки трейт– это тоже класс. Полное имя трейта будет передано в функцию автозагрузки.*

ТРЕЙТЫ





CRUD И ШАБЛОН ACTIVERECORD

Active Record - это архитектурный паттерн «Активная запись».

- Записи в БД соответствует объект в языке программирования
- Запись в базе данных может «сама себя» сохранить и удалить, используя методы объекта

Например:

```
$user = User::findById(1);
$user->password = 'password';
$user->save();
```

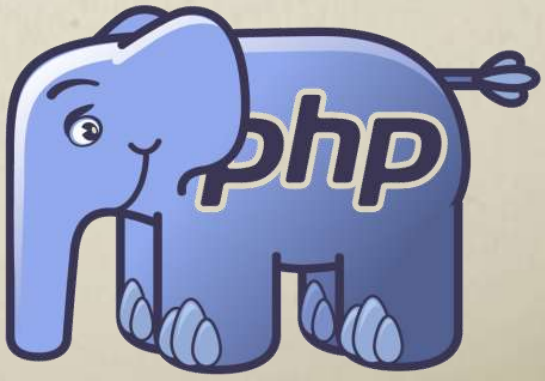
или

```
$user = new User;
$user->name = 'Василий';
$user->email = 'vasya@test.com';
$user->save();
```

или

```
$user = User::findByEmail('1@xxx.ru');
$user->delete()
```

ACTIVE RECORD



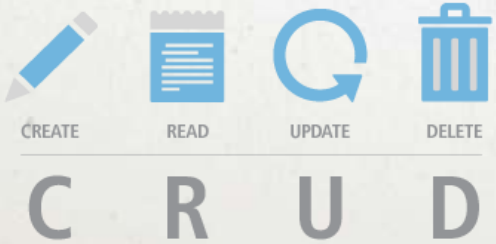
CRUD - это сокращение от Create, Read, Update, Delete – 4 основные операции с данными

Read мы уже умеем.

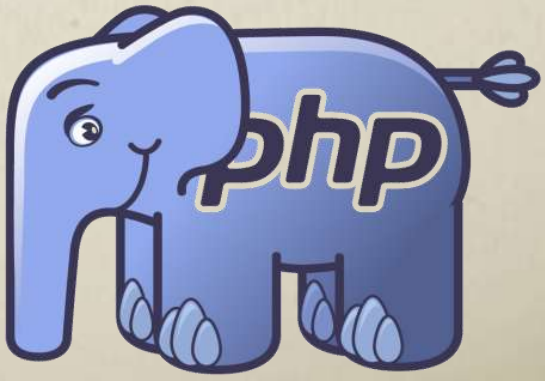
Для **Create** давайте реализуем метод insert() в наших моделях:

- Вставлять в базу можно только «новую модель», а не ту, что мы уже получили от БД
- У новой модели еще нет первичного ключа
- А вот после успешной вставки он должен появиться
- И нужно как-то определить список полей для записи в БД…

Аналогично попробуем сделать метод update() для случая, когда мы обновляем уже существующие данные.



CRUD



ДО ВСТРЕЧИ НА СЛЕДУЮЩЕМ УРОКЕ!

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

ProfIT