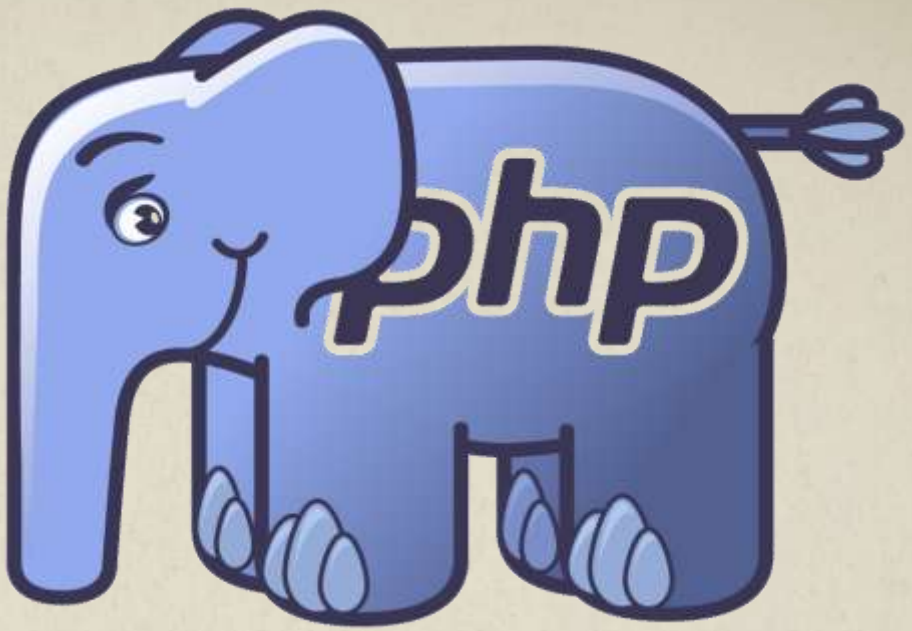
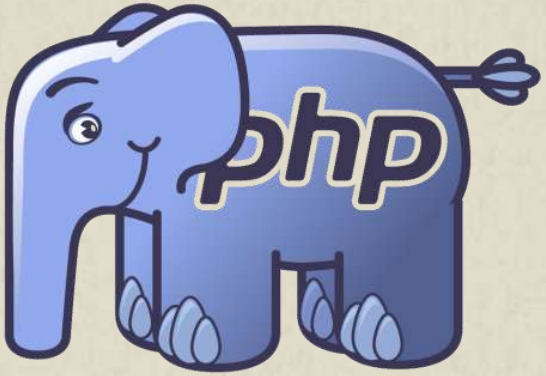


PHP: ВВЕДЕНИЕ В ПРОФЕССИЮ

УРОК 6. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД.



КЛАСС, КАК ТИП

Класс – это тип объектов.

- Все объекты в PHP имеют тип object. Это легко проверить:

```
$obj = new SomeClass;
echo gettype( $obj ); // object
```

- Однако, каждый класс можно рассматривать тоже как некий тип данных!

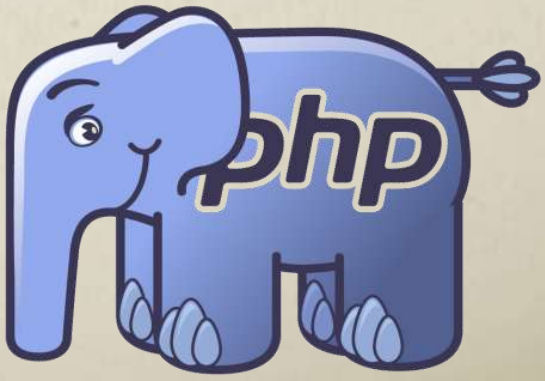
```
$user = new User;
echo get_class($user); // User
```

- В PHP существует специальный оператор для проверки того, является ли объект объектом заданного класса:

```
if ($obj instanceof User) {
   ...
}
```

- Интересно, что такая проверка даст `true`, если имя класса, которое вы укажете, будет одним из предков класса данного объекта

КЛАССЫ, КАК ТИПЫ



Класс – это тип объектов.

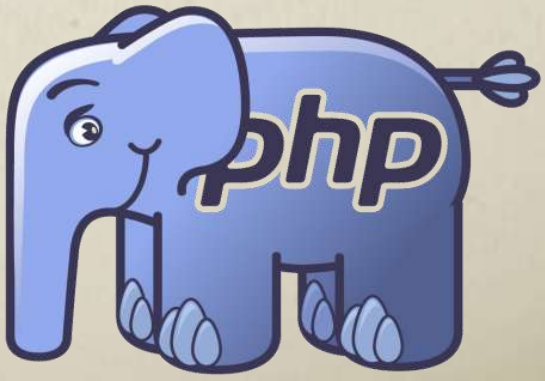
- Есть еще одно отличное применение классам, как типам. Это «тайп-хинтинг»:

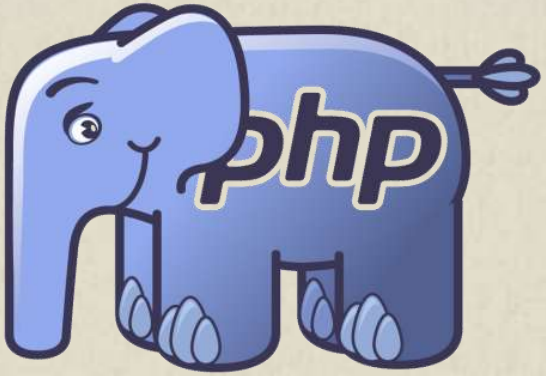
```
function sendMail(User $user, $msg)
{
  ...
}

$user = new User(...);
sendMail($user, 'hello!');
```

- Вызов такой функции будет успешен, если аргумент $user – объект класса User (или класса-наследника)
- Если же мы попытаемся передать что-то другое – возникнет фатальная ошибка!
- Тайп-хинтинг позволяет нам контролировать ход выполнения программы и использовать принципы ООП

КЛАССЫ, КАК ТИПЫ





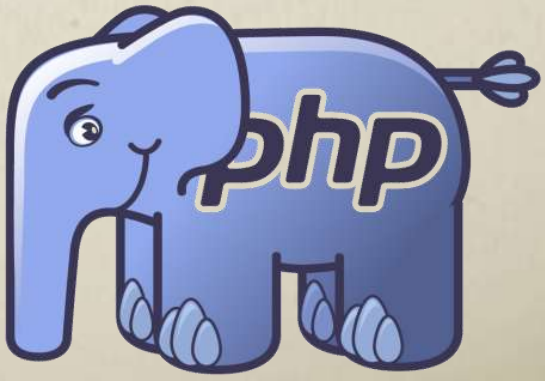
МОДЕЛИ ДАННЫХ. ORM.

Объекты принято использовать в качестве МОДЕЛЕЙ данных

- **ORM – Object Relational Mapping**
  или, иначе говоря, принцип отображения объектов реального мира (и их связей) на объекты вашего языка программирования
- ООП прекрасно подходит для реализации этого шаблона проектирования.
  - Класс описывает какие объекты данных у нас могут быть
  - Сами данные представлены объектами заданных классов
  - И мы сразу может определить «поведение данных» в виде методов этих объектов

```
class GuestBookRecord {
  public function __construct($msg) {
    ...
  }
  public function getText() {
    ...
  }
}
```

МОДЕЛИ



Объекты принято использовать в качестве МОДЕЛЕЙ данных

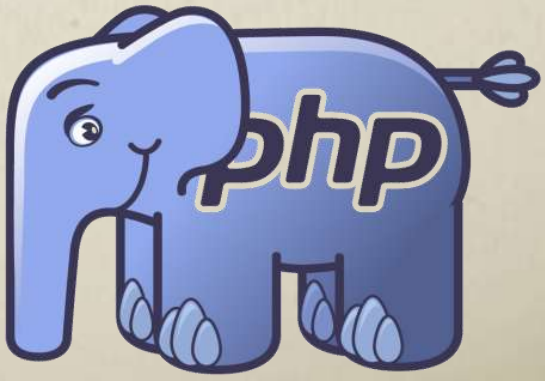
- **ORM – Object Relational Mapping**
  или, иначе говоря, принцип отображения объектов реального мира (и их связей) на объекты вашего языка программирования
- Методы, возвращающие данные, должны возвращать массивы объектов нужного класса
- Методы, принимающие данные – принимать объекты нужного класса

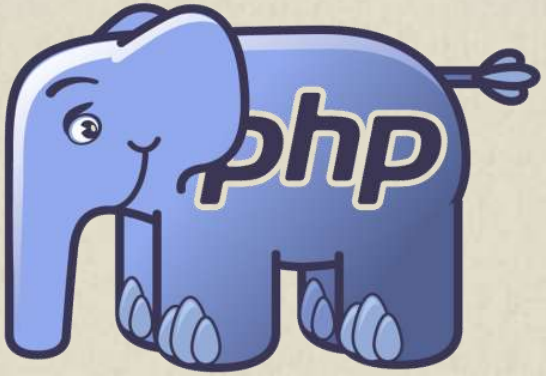
```
function add(GuestBookRecord $rec) {
   ...
}
```

- Допускается создание «мета-объектов», например класса `GuestBook`
- И, разумеется, связь между объектами!

NB. Не бойтесь «копирования»! Объекты передаются по ссылкам!

МОДЕЛИ





БУФЕР ВЫВОДА

ВСЁ, что мы выводим – уже нельзя вернуть назад...

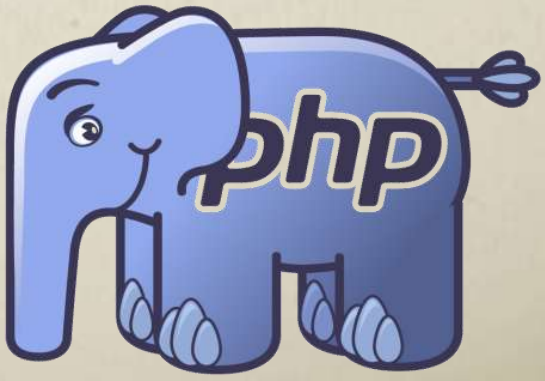
- Делаем, например, вот так:
```
echo 'Hello!';
```
- Или так:
```
?>
<strong>Автор:</strong>
<?php
...
```

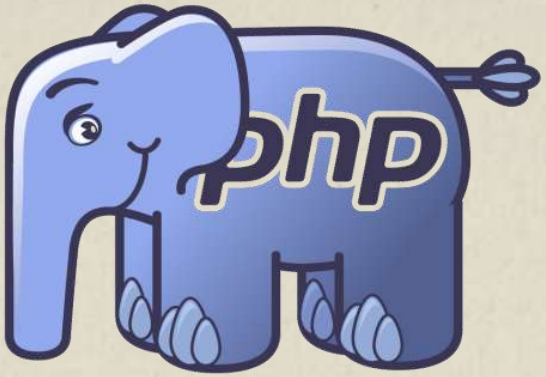
В таком случае всё, что мы выводим, непосредственно отправляется сразу клиенту.

Однако, в PHP есть способ взять вывод в «буфер», чтобы потом с ним работать:

```
ob_start();
echo 'Hello!';
$out = ob_get_contents();
ob_end_clean();
```

БУФЕР ВЫВОДА





ОТДЕЛЕНИЕ ПРЕДСТАВЛЕНИЯ ОТ КОДА

ПРАВИЛЬНО разделять представление и бизнес-логику

- Сверстайте шаблон вашей страницы. Обычный HTML
- Определите, какие данные вы будете ей передавать
- Подготовьте данные, получив их от ваших моделей
- Передайте (в простейшем случае – просто include!)
- В шаблоне реализуйте ЛОГИКУ ПРЕДСТАВЛЕНИЯ, то есть то, как будут отображаться ваши данные
- И, наконец, создайте специальный объект, который будет управлять представлением:
  - Хранить в себе данные для него
  - Отображать заданный шаблон с этими данными

ПРЕДСТАВЛЕНИЕ