Запись на курсы по HTML, CSS, JavaScript, PHP, фреймворкам и CMS,
а также: помощь в поиске работы и заказов, стажировка на реальных проектах→

Бесплатные курсы по React для новичков. Начало 4-го ноября→ Конкурс CSS картинок. Тема: Хэллоуин. Призовой фонд: 100$. Подробности→

Объект со статическими свойствами и методами

Вы уже знаете, что статические свойства и методы можно использовать, не создавая объект класса. На самом деле, однако, класс может содержать как статические свойства и методы, так и обычные.

Давайте посмотрим, как с этим работать и какие преимущества это дает. Пусть у нас есть класс **Test** одновременно и со статическим свойством, и с обычным:

```
<?php
  class Test
  {
    public static $staticProperty; // публичное статическое сво
      йство
     public $usualProperty; // обычное свойство
  }
?>
```

Давайте, к примеру, поработаем с его обычным свойством:

```
<?php
  $test = new Test; // создаем объект класса

  $test->usualProperty = 'usual'; // записываем зна
    чение
  e cho $test->usualProperty; // выведет 'usu
    al'
?>
```

А теперь используем статическое свойство, не создавая объект этого класса:

```
<?php
  Test::$staticProperty = 'static'; // записываем зна
    чение
  e cho Test::$staticProperty; // выведет 'sta
    tic'
?>
```

На самом деле, если у нас есть переменная с объектом класса, то у этой переменной также будет доступно статическое свойство:

```
<?php
  $test = new Test; // создаем объект класса

  $test::$staticProperty = 'static'; // записываем зна
    чение
  e cho $test::$staticProperty; // выведет 'sta
    tic'
?>
```

Разницы нет - мы к одному и тому же статическому свойству можем обращаться и так, и так. Вот пример, иллюстрирующий это:

```php
<?php
  // Записываем значение еще ДО создания объ
    екта:
  T est::$staticProperty = 'static';

  // Создаем объект класса:
  $test = new Test;

  // Выводим статическое свойство:
  echo $test::$staticProperty; // выведет 'sta
    tic'
?>
```

Вот еще пример:

```php
<?php
  // Создаем объект класса:
  $test = new Test;

  // Записываем значение в статическое сво
    йство:
  $ test::$staticProperty  = 'static';

  // Выводим значение, обратившись к классу:
  echo Test::$staticProperty; // выведет 'sta
    tic'

  // Выводим значение, обратившись к объекту класса:
  echo $test::$staticProperty; // выведет 'sta
    tic'
?>
```

Несколько объектов

Статические свойства принадлежат не какому-то объекту класса, а самому классу, хотя объекты класса и имеют доступ к этим свойствам.

На практике это означает то, что если у нас есть несколько объектов класса - статические свойства у них будут общие. То есть, если в одном объекте поменять значение статического свойства - изменения произойдут во всех объектах.

Давайте посмотрим на примере:

```php
<?php
  $test1 = new Test; // первый объект
  $test2 = new Test; // второй объект

  $test1::$staticProperty = 'static'; // зап
    ишем значение, используя первый объект

  echo $test1::$staticProperty; // выведет 'sta
    tic'
  e cho $test2::$staticProperty; //!! также выведет 'st
    atic'
?>
```

Статические методы и $this

Пусть у нас есть класс **Test** с двумя свойствами, статическим и обычным:

```php
<?php
  class Test
  {
    public static $staticProperty = 'static'; // ста
```

```
    тическое свойство
    public $usualProperty = 'usual'; // обычное сво
    йство
  }
?>
```

Давайте выведем значения этих свойств в обычном (нестатическом) методе **method**:

```
<?php
  class Test
  {
    public static $staticProperty = 'static'; // ста
    тическое свойство
     public $usualProperty = 'usual'; // обычное сво
    йство

    // Обычный метод:
    public function method()
    {
      var_dump(self::$staticProperty); // выведет 'sta
      tic'
       var_dump($this->usualProperty); // выведет 'usu
      al'
    }
  }

  $test = new Test;
  $test->method(); // обычный метод - вызываем чер
  ез ->
?>
```

Из примера видно, что в обычном методе доступны как статические, так и обычные свойства (и методы).

Пусть теперь наш метод **method** будет статическим. В этом случае он сможет обратиться с статическим методом и свойствам, но к обычным - нет.

Почему: потому что внутри статических методов недоступен **$this**. Это происходит из-за того, что статические методы могут вызываться вне контекста объекта, просто обращаясь к имени класса.

А ведь **$this** внутри класса как раз-таки ссылается на объект этого класса. Нет объекта - **$this** ни на что не ссылается.

Убедимся в этом: переделаем наш метод на статический - теперь обращение к обычному свойству внутри нашего метода будет выдавать ошибку:

```
<?php
  class Test
  {
    public static $staticProperty = 'static'; // ста
    тическое свойство
     public $usualProperty = 'usual'; // обычное сво
    йство

    // Переделали на статический метод:
    public static function method()
    {
      var_dump(self::$staticProperty); // выведет 'sta
      tic'
      var_dump($this->usualProperty); //!! выдаст ошибку
    }
  }

  $test = new Test;
  $test::method(); // статический метод - вызываем чер
  ез ::
?>
```

Применение

Пусть у нас есть вот такой класс **User**:

```php
<?php
  class User
  {
    public $name;

    public function __construct($name)
    {
      $this->name = $name;
    }
  }
?>
```

Давайте сделаем так, чтобы этот класс подсчитывал количество своих объектов.

Для этого сделаем статическое свойство **count**. Изначально запишем в него значение 0, а при создании каждого нового объекта будем увеличивать это значение на 1.

Будем увеличивать значение нашего счетчика в конструкторе объекта:

```php
<?php
  class User
  {
    public static $count = 0; // счетчик объ
      ектов
     public $name;

    public function __construct($name)
    {
      $this->name = $name;

      // Увеличиваем счетчик при создании объекта:
      self::$count++;
    }
  }
?>
```

Проверим, что все работает:

```php
<?php
  $user1 = new User('user1'); // создаем пер
    вый объект класса
  e cho User::$count; //выведет 1

  $user2 = new User('user2'); // создаем вто
    рой объект класса
  e cho User::$count; //выведет 2
?>
```

Задача 31.1

Не подсматривая в мой код реализуйте такой же класс **User**, подсчитывающий количество своих объектов.

Улучшим наш код

Не очень хорошо то, что наш счетчик публичный - его случайно можно изменить снаружи класса.

Давайте сделаем его доступным только для чтения. Для этого объявим его приватным и сделаем для него статический метод-геттер **getCount**:

```php
<?php
  class User
  {
    private static $count = 0;
    public $name;

    public function __construct($name)
    {
      $this->name = $name;

      // Увеличиваем счетчик при создании объекта:
      self::$count++;
    }

    // Метод, возвращающий значение счетчика:
    public static function getCount()
    {
      // Выводим значение счетчика:
      return self::$count;
    }
  }
?>
```

Проверим:

```php
<?php
  $user1 = new User('user1'); // создаем пер
    вый объект класса
  e cho User::getCount(); //выведет 1

  $user2 = new User('user2'); // создаем вто
    рой объект класса
  e cho User::getCount(); //выведет 2
?>
```

Задача 31.2

Самостоятельно переделайте код вашего класса **User** в соответствии с теоретической частью урока.