Магический метод __set

Магический метод **__set** вызывается при попытке изменить значение несуществующего или скрытого свойства.

В качестве параметров он принимает имя свойства и значение, которое ему пытаются присвоить.

Давайте посмотрим на практическом примере. Пусть у нас дан вот такой класс **Test**:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;
  }
?>
```

Давайте сделаем в этом классе магический метод **__set**, который с помощью функции **var_dump** будет выводить имя свойства, к которому произошло обращение, и значение, которое этому свойству пытаются установить:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      var_dump($property . ' ' .$value);
    }
  }
?>
```

> Параметры **$property** и **$value** могут иметь любое название. Главное, чтобы они вообще были. В эти параметры PHP будет передавать имя свойства (в первый параметр) и значение (во второй параметр).

Проверим работу нашего класса:

```php
<?php
  $test = new Test;
  $test->prop = 'value'; // var_dump метода __s
    et выведет 'prop value'
?>
```

Давайте теперь будем устанавливать значение свойству, имя которого хранится в переменной **$property**:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      $this->$property = $value; // устанавливаем зна
        чение
     }

```

```
    }
?>
```

Теперь мы сможем записывать в приватные свойства снаружи класса:

```php
<?php
  $test = new Test;

  $test->prop1 = 1; // запишем 1
  $test->prop2 = 2; // запишем 2
?>
```

Записывать мы можем, однако, проверить, записалось ли туда что-то - нет, так как свойства приватные.

Можно сделать геттер для этих свойств или просто воспользоваться магическим методом **__get**. Воспользуемся вторым вариантом:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      $this->$property = $value;
    }

    // Магический геттер свойств:
    public function __get($property)
    {
      return $this->$property;
    }
  }
?>
```

Вот теперь мы можем проверить работу нашего класса. Проверим:

```php
<?php
  $test = new Test;

  $test->prop1 = 1; // запишем 1
  $test->prop2 = 2; // запишем 2

  echo $test->prop1; // выведет 1
  echo $test->prop2; // выведет 2
?>
```

На самом деле, конечно же, не стоит разрешать всем подряд записывать в приватные свойства, иначе пропадает суть этих приватных свойств (проще сделать их публичными и все).

Поэтому данный метод следует применять только тогда, когда в этом действительно есть необходимость. Ниже мы еще рассмотрим примеры удачного применения.

Несуществующее свойство

Давайте попробуем записать данные в несуществующее свойство - это будет работать:

```php
<?php
  $test = new Test;

  $test->prop3 = 3; // запишем 3
  echo $test->prop3; // выведет 3
?>
```

Пусть мы не хотим разрешать записывать в несуществующие свойства. И, вообще, хотим разрешить запись только в свойства **prop1** и **prop2**.

Это легко сделать - достаточно в методе **__set** добавить соответствующее условие:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      // Напишем условие:
      if ($property == 'prop1' or $property ==
        'prop2') {
        $this->$property = $value;
      }
    }

    public function __get($property)
    {
      return $this->$property;
    }
  }
?>
```

Если таких свойств будет много, то не очень удобно перечислять их все в условии.

Давайте запишем разрешенные для записи свойства в массив и будем проверять наличие свойства в этом массиве с помощью функции **in_array**:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      $properties = ['prop1', 'prop2']; // раз
        решенные свойства

      if (in_array($property, $properties)) {
        $this->$property = $value;
      }
    }

    public function __get($property)
    {
      return $this->$property;
    }
  }
?>
```

Проверка при записи

Давайте будем проверять значения свойств на соответствие определенному условию:

```php
<?php
  class Test
  {
    private $prop1;
    private $prop2;

    public function __set($property, $value)
    {
      switch($property) {
        case 'prop1':
          // Если prop1 от 0 до 10:
          if ($value > 0 and $value < 10) {
            $this->$property = $value;
          }
```

```
        break;
        case 'prop2':
          // Если prop2 от 10 до 20:
          if ($value > 10 and $value < 20) {
            $this->$property = $value;
          }
        break;
        default:
          // Такого свойства нет
        break;
      }
    }

    public function __get($property)
    {
      return $this->$property;
    }
  }
?>
```

Применение

Практическое применение метода **__set** вы изучите самостоятельно, решив вот такую задачу:

Задача 55.1

Пусть дан вот такой класс **User** с геттерами и сеттерами свойств:

```
<?php
  class User
  {
    private $name;
    private $age;

    public function getName()
    {
      return $this->name;
    }

    public function setName($name)
    {
      if ($name != '') { // проверяем имя на
        непустоту
        $this->name = $name;
      }
    }

    public function getAge()
    {
      return $this->age;
    }

    public function setAge($age)
    {
      if ($age >= 0 and $age <= 70) { // проверяем воз
        раст
        $this->age = $age;
      }
    }
  }
?>
```

Переделайте код этого класса так, чтобы вместо геттеров и сеттеров использовались магический методы **__get** и **__set**.

Возможно, после переделки код нашего класса стал немного запутаннее, но иногда, когда проверка присваиваемых свойствам значений одинакова или чем-то похожа такой прием может немного сократить и улучшить ваш код.