Магический метод __get

Следующий магический метод, который мы с вами разберем, называется **__get**. Этот метод срабатывает при попытке прочитать значение несуществующего или скрытого (то есть private или protected) свойства.

Если реализовать метод **__get** в каком-нибудь классе, то все обращения к несуществующим или скрытым свойствам будут обрабатываться этим методом.

При этом PHP автоматически будет передавать имя запрошенного свойства в первый параметр этого метода, а возвращаемое этим методом значение будет воспринято как значение свойства, к которому произошло обращение.

Скорее всего пока не очень понятно, как это работает, поэтому давайте посмотрим на практическом примере.

Пусть у нас есть вот такой класс **Test** с приватным и публичным свойствами:

```php
<?php
  class Test
  {
    public $prop1 = 1; // публичное свойство
    private $prop2 = 2; // приватное свойство
  }
?>
```

Давайте добавим в наш класс магический метод **__get**, который для начала будет просто возвращать имя свойства, к которому произошло обращение:

```php
<?php
  class Test
  {
    public $prop1 = 1;
    private $prop2 = 2;

    public function __get($property)
    {
      return $property; // просто вернем имя свойства
    }
  }
?>
```

> Имя параметра (в нашем случае **$property**) может быть любым. Просто в этот параметр сам PHP будет передавать имя свойства, к которому произошло обращение.

Давайте проверим работу созданного магического метода. Обратимся к трем типам свойств: к публичному свойству, к приватному и к несуществующему:

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

  // Обращаемся к публичному свойству:
  echo $test->prop1; // выведет 1 - то есть значение сво
    йства

  // Обращаемся к приватному свойству:
  echo $test->prop2; // выведет 'prop2' - имя сво
    йства

  // Обращаемся к несуществующему свойству:
  echo $test->prop3; // выведет 'prop3' - имя сво
```

```
        йства
12  ?>
```

Как вы видите, наш магический метод реагирует на обращение к приватным и несуществующим свойствам, но игнорирует обращение к публичным - они работают так, как и работали раньше.

Применение: свойства только для чтения

Пусть теперь в нашем классе все свойства приватные:

```php
<?php
  class Test
  {
    private $prop1 = 1;
    private $prop2 = 2;
  }
?>
```

Давайте сделаем так, чтобы эти свойства во внешнем мире были доступны только для чтения.

Ранее мы такое уже делали, создавая геттеры для каждого свойства и не создавая сеттеры.

Давайте теперь для решения этой задачи воспользуемся магическим методом **__get**. Будем возвращать в нем значение запрошенного свойства.

Как это сделать: имя запрошенного свойства попадает в параметр метода **__get**, в нашем случае **$property**.

Это значит, что мы можем прочитать свойство, имя которого хранится в переменной, вот так: **$this->$property** (имя свойства будет переменной, то есть с долларом вначале, мы это проходили в предыдущих уроках).

Давайте сделаем описанный метод **__get**:

```php
<?php
  class Test
  {
    private $prop1 = 1;
    private $prop2 = 2;

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

Воспользуемся им для чтения свойств:

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

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

Попытка записать что-то в свойство приведет к ошибке:

```php
<?php
  $test = new Test;
  $test->prop1 = 2; // выдаст ошибку
?>
```

Это именно то, что нам нужно: свойство можно прочитывать, но нельзя записывать.

Попытка прочитать несуществующее свойство выдаст ошибку (уровня notice):

```php
<?php
  $test = new Test;
  echo $test->prop3; // выдаст ошибку
?>
```

Обратите также внимание на следующий нюанс: когда мы делали свойства только для чтения старым способом, то для того, чтобы прочитать свойство, мы использовали метод-геттер.

В новом способе мы будем обращаться именно к свойствам, будто они публичные. Но записать в них не сможем, будто они приватные.

Задача 54.1

Пусть дан вот такой класс **User**, свойства которого доступны только для чтения с помощью геттеров:

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

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

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

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

Переделайте код этого класса так, чтобы вместо геттеров использовался магический метод **__get**.

Несуществующее свойство

В примере выше мы применяли магию метода **__get** для отлавливания обращения к приватным свойствам.

На самом деле этот метод также может быть полезен для отлавливания обращений к несуществующим свойствам.

Посмотрим на практическом примере.

Пусть у нас есть класс **User** с фамилией, именем и отчеством, являющимися публичными свойствами:

```php
<?php
  class User
  {
    public $surname; // фамилия
    public $name; // имя
    public $patronymic; // отчество
  }

  $user = new User;

  $user->surname = 'Иванов';
  $user->name = 'Иван';
  $user->patronymic = 'Иванович';
?>
```

Давайте сделаем так, чтобы объект класса вел себя так, будто у него также есть свойство **fullname**, выводящее ФИО юзера:

```php
<?php
  $user = new User;

```

```php
  $user->surname = 'Иванов';
  $user->name = 'Иван';
  $user->patronymic = 'Иванович';

  // Выведет 'Иванов Иван Иванович':
  echo $user->fullname; // это пока не работает, явл
    яется нашей целью
?>
```

Используем для этого наш магический метод **__get**:

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

    // Используем метод-перехватчик:
    public function __get($property)
    {
      // Если идет обращение к свойству fullname:
      if ($property == 'fullname') {
        return $this->surname . ' ' . $this->nam
    e . ' ' . $this->patronymic;
      }
    }
  }
?>
```

Проверим:

```php
<?php
  $user = new User;

  $user->surname = 'Иванов';
  $user->name = 'Иван';
  $user->patronymic = 'Иванович';

  echo $user->fullname; // выведет 'Иванов Ива
    н Иванович'
?>
```

Получается, что с помощью **__get** мы создали в классе виртуальное свойство: в классе его нет, но прочитать его можно.

Кстати, записать в такое свойство будет нельзя, так как в реальности его не существует в нашем классе. То есть это свойство только для чтения.

Задача 54.2

Сделайте класс **Date** с публичными свойствами **year** (год), **month** (месяц) и **day** (день).

С помощью магии сделайте свойство **weekDay**, которое будет возвращать день недели, соответствующий дате.