Класс Input

Давайте теперь сделаем класс **Input** для работы с инпутами. Вот готовая реализация этого класса:

```php
<?php
  class Input extends Tag
  {
    public function __construct()
    {
      parent::__construct('input');
    }
  }
?>
```

Давайте используем новый класс **Input** вместе с уже созданным нами классом **Form**:

```php
<?php
  $form = (new Form)->setAttrs(['action' => '',
    'method' => 'GET']);

  echo $form->open();
    echo (new Input)->setAttr('name', 'year')->
      open();
     echo (new Input)->setAttr('type', 'submit')->
      open();
  e cho $form->close();
?>
```

В результате получится следующая форма:

```html
<form action="" method="GET">
  <input name="year">
  <input type="submit">
</form>
```

Задача 72.1

Скопируйте себе мой код класса **Input**. Скопируйте пример формы. Запустите его в браузере, убедитесь, что форма действительно появляется. Вбейте какие-нибудь данные в инпут и нажмите на кнопку отправки. Убедитесь, что форма действительно рабочая.

Убираем open

Давайте уберем метод **open**, добавив магию метода **__toString**:

```php
<?php
  class Input extends Tag
  {
    public function __construct()
    {
      parent::__construct('input');
```

```
    }

    public function __toString()
    {
      return parent::open();
    }
  }
?>
```

Задача 72.2

Перепишите приведенный ниже код с учетом нашей правки:

```
<?php
  $form = (new Form)->setAttrs(['action' => '',
    'method' => 'GET']);

  echo $form->open();

    echo (new Input)->setAttr('name', 'year')->
      open();
     echo (new Input)->setAttr('type', 'submit')->
       open();
   e cho $form->close();
?>
```

Некоторые замечания

Пока наш PHP код формы получается длиннее соответствующего HTML кода. Возникает вопрос: зачем нам PHP вариант, если все намного короче можно написать на HTML?

Все дело в том, что наша PHP реализация дает нам дополнительные возможности. Например, мы можем сделать так, чтобы данные из инпутов не исчезали после отправки формы.

И теперь преимущество в длине кода будет уже на стороне PHP варианта - длина HTML варианта существенно вырастет, а длина PHP варианта останется неизменной.

Реализация сохранения значений после отправки

Как вы должны знать, значение инпута задается в атрибуте **value**. Нам необходимо сделать следующее: если форма была отправлена, то в **value** инпута следует записать то значение, которое было в нем в момент отправки.

Каким образом получить это значение? Пусть имя инпута было **year**. В этом случае после отправки формы в **$_REQUEST['year']** и будет находится нужное нам значение.

В нашем случае имя инпута хранится в родительском классе **Tag** в приватном свойстве **$attrs**. Наш класс **Input** как потомок класса **Tag** может получить доступ к любому атрибуту с помощью геттера, вот так: *$inputName = $this->getAttr('name')*.

Давайте получим имя нашего инпута, достанем значение из **$_REQUEST** по этому имени и запишем в **value** инпута:

```
<?php
  class Input extends Tag
  {
    public function __construct()
    {
      parent::__construct('input');
    }

    // Переопределяем метод родителя:
    public function open()
    {
      $inputName = $this->getAttr('name');
        // имя инпута
```

```
      $value = $_REQUEST[$inputName]; // пол
        учаем значение инпута по его имени

      $this->setAttr('value', $value); // записываем в v
        alue инпута

      return parent::open(); // вызываем метод ope
        n родителя
    }

  public function __toString()
  {
    return $this->open(); //!! здесь теперь не
      метод родителя, а наш
    }
  }
?>
```

Приведенный выше код пока не очень корректный: он не учитывает того, что отправки формы могло еще и не быть. Давайте учтем:

```
<?php
  // Пока не запускайте этот код, есть нюансы...

  class Input extends Tag
  {
    public function __construct()
    {
      parent::__construct('input');
    }

    // Переопределяем метод родителя:
    public function open()
    {
      $inputName = $this->getAttr('name');
        // имя инпута

      // Если форма была отправлена и есть дан
        ные с именем нашего инпута:
      if (isset($_REQUEST[$inputName])) {
        $value = $_REQUEST[$inputName]; // пол
          учаем значение из $_REQUEST
        $this->setAttr('value', $value); // записываем в v
          alue инпута
      }

      return parent::open(); // вызываем метод ope
        n родителя
    }

  public function __toString()
  {
    return $this->open();
  }
  }
?>
```

Вот теперь наш код уже достаточно рабочий, но есть пару нюансов.

Что будет, если у нашего инпута по умолчанию уже есть какое-то значение (например, текущий год), например, вот так:

```
<?php
  echo (new Input)
    ->setAttr('name', 'year')
    ->setAttr('value', date('Y')); // значение по
```

```
        умолчанию
?>
```

В этом случае **в теории** по заходу на страницу мы должны увидеть текущий год, но при желании можем его поменять, нажать на кнопку отправки - и после обновления страницы в инпуте будет тот год, который и был отправлен.

На самом деле наш код так и работает! Если была отправка формы, то мы попадем в это условие *if (isset($_REQUEST[$inputName]))* и там изменим **value** инпута, но если отправки не было - **value** инпута останется таким, как и было задано.

Второй нюанс: есть инпуты, у которых нет атрибута **name**. Например, кнопка отправки формы. В этом случае с такими инпутами ничего и не надо делать. Учтем это дополнительным условием:

```
<?php
  class Input extends Tag
  {
    public function __construct()
    {
      parent::__construct('input');
    }

    public function open()
    {
      $inputName = $this->getAttr('name');

      // Если атрибут name задан у инпута:
      if ($inputName) {
        if (isset($_REQUEST[$inputName])) {
          $value = $_REQUEST[$inputName];
          $this->setAttr('value', $value);
        }
      }

      return parent::open();
    }

    public function __toString()
    {
      return $this->open();
    }
  }
?>
```

Ну вот теперь можно пробовать. Можете запустить приведенный ниже код формы, вбить что-то в инпут и нажать на кнопку отправки - после обновления страницы данные из инпута не исчезнут:

```
<?php
  $form = (new Form)->setAttrs(['action' => '',
    'method' => 'GET']);

  echo $form->open();
    echo (new Input)->setAttr('name', 'year');
    echo (new Input)->setAttr('type', 'submit');
  echo $form->close();
?>
```

А вот пример, когда в инпуте уже есть какое-то значение по умолчанию:

```
<?php
  $form = (new Form)->setAttrs(['action' => '',
    'method' => 'GET']);

  echo $form->open();
    echo (new Input)->setAttr('name', 'year')->
      setAttr('value', date('Y'));
     echo (new Input)->setAttr('type', 'submit');

```

```
    echo $form->close();
8 ?>
```

Задача 72.3

Реализуйте самостоятельно сохранение значений инпутов после отправки. Проверьте работу данного механизма.

Задача 72.4

С помощью созданного класса сделайте форму с 5-ю инпутами. Пусть в каждый инпут можно ввести число. Сделайте так, чтобы после отправки на экран выводилась сумма этих чисел, а введенные значения не пропадали из инпутов.