Разработка многостраничного сайта на PHP

ЗАНЯТИЕ № 10,11,12
ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема занятия – введение в объектно-ориентированное программирование

Цель занятия –

Продолжить изучать объектно-ориентированного программирования на практике

Актуализация

На прошлом занятии мы учились на практике :

- создавать классы,
- решали разные задачи, связанные с ООП

Практическое занятие

Сегодня на занятии мы продолжим составлять классы, используя методы ООП в PHP.

Задача № 1

Создайте класс **Form** - оболочку для создания форм. Он должен иметь методы input, submit, password, textarea, open, close. Каждый метод принимает массив атрибутов.

На следующем слайде посмотрим на примеры использования методов.

Примеры использования методов

```php
<?php
    echo $form->input(['type'=>'text', 'value'=>'!!!']);
    //Код выше выведет <input type="text" value="!!!">

    echo $form->password(['value'=>'!!!']);
    //Код выше выведет <input type="password" value="!!!">

    echo $form->submit(['value'=>'go']);
    //Код выше выведет <input type="submit" value="go">

    echo $form->textarea(['placeholder'=>'123', 'value'=>'!!!']);
    //Код выше выведет <textarea placeholder="123">!!!</textarea>

    echo $form->open(['action'=>'index.php', 'method'=>'POST']);
    //Код выше выведет <form action="index.php" method="POST">

    echo $form->close();
    //Код выше выведет </form>
?>
```

Задача №1 (продолжение)

Передаваемые атрибуты могут быть любыми:

```php
<?php
    echo $form->input(['type'=>'text', 'value'=>'!!!', 'class'=>'ggg']);
    //Код выше выведет <input type="text" value="!!!" class="ggg">
?>
```

Для решения задачи сделайте **private** метод, который параметром будет принимать массив, например, *['type'=>'text', 'value'=>'!!!']* и делать из него строку с атрибутами, в нашем случае *type="text" value="!!!"*.

Задача №1 (продолжение)

Пример создания формы с помощью нашего класса:

```php
<?php
    echo $form->open(['action'=>'index.php', 'method'=>'POST']);
    echo $form->input(['type'=>'text', 'placeholder'=>'Ваше имя', 'name'=>'name']);
    echo $form->password(['placeholder'=>'Ваш пароль', 'name'=>'pass']);
    echo $form->submit(['value'=>'Отправить']);
    echo $form->close();
?>
```

В результате получится следующая форма:

```html
<form action="index.php" method="POST">
    <input type="text" placeholder="Ваше имя" name="name">
    <input type="text" placeholder="Ваш пароль" name="pass">
</form>
```

Задача № 2

Создайте класс **SmartForm**, который будет наследовать от **Form** из предыдущей задачи и сохранять значения input и textarea после отправки.

То есть если мы сделали форму, нажали на кнопку отправки - то значения из input не должны пропасть. Мало ли что-то пойдет не так, например, форма некорректно заполнена, а введенные данные из нее пропали и их следует вводить заново. Этого следует избегать.

Задача № 3

Создайте класс **Cookie** - оболочку над работой с куками. Класс должен иметь следующие методы: установка куки *set(имя куки, ее значение)*, получение куки *get(имя куки)*, удаление куки *del(имя куки)*.

Задача № 4

Создайте класс **Session** - оболочку над сессиями. Он должен иметь следующие методы: создать переменную сессии, получить переменную, удалить переменную сессии, проверить наличие переменной сессии.

Сессия должна стартовать (session_start) в методе **__construct**.

Задача № 5

Реализуйте класс **Flash**, который будет использовать внутри себя класс Session из предыдущей задачи (именно использовать, а не наследовать).

Этот класс будет использоваться для сохранения сообщений в сессию и вывода их из сессии. Зачем это нужно: такой класс часто используется для форм. Например на одной странице пользователь отправляет форму, мы сохраняем в сессию сообщение об успешной отправке, редиректим пользователя на другую страницу и там показываем сообщение из сессии.

Класс должен иметь да метода - **setMessage**, который сохраняет сообщение в сессию и **getMessage**, который получает сообщение из сессии.

Задача № 6

Создайте класс-оболочку **Db** над базами данных. Класс должен иметь следующие методы: получение данных, удаление данных, редактирование данных, подсчет данных, очистка таблицы, очистка таблиц.

Задача № 7

Создайте класс **Log** для ведения логов. Этот класс должен иметь следующие методы: сохранить в лог, получить последние N записей, очистить таблицу с логами.

Класс **Log** должен использовать класс **Db** из предыдущей задачи (именно использовать, а не наследовать).

Рефлексия

Сегодня на практическом занятии мы отработали те теоретические навыки, которые мы получили на прошлом занятии.

Ответьте, пожалуйста, на несколько вопросов:

1. Какая задача была самая интересная?
2. Какая задача вам показалась наиболее легкой и почему?
3. Какая задача вам показалась наиболее тяжелой и почему?
4. Остались ли у вас какие-нибудь вопросы после нашего занятия?

Спасибо за внимание