УДК 004

Принципы объектно-ориентированного программирования в PHP

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматриваются основные понятия объектно-ориентированного программирования (далее ООП). Разобран пример с использованием основных принципов подхода для реализации простой авторизации.
Ключевые слова: ООП, программирование, PHP.

Principles of object-oriented programming in PHP

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article discusses the basic concepts of OOP. An example using the basic principles of the approach for the implementation of simple authorization
Keywords: OOP, programming, PHP.

Объектно-ориентированное программирование (ООП) - это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса (типа особого вида), а классы образуют иерархию, основанную на принципах наследуемости [4].

Целью исследования является рассмотрение основных принципов объектно-ориентированное программирование и реализация их на практике.

Объектно-ориентированные программы более просты и мобильны, их легче модифицировать и сопровождать, чем их «традиционных» собратьев. Кроме того, похоже, сама идея объектной ориентированности при грамотном ее использовании позволяет программе быть даже более защищенной от различного рода ошибок, чем это задумывал программист в момент работы над ней. PHP до недавнего времени обеспечивал лишь некоторую поддержку ООП. Однако, после выхода PHP5 поддержка ООП в PHP стала практически полной.

Технология ООП обладает тремя главными преимуществами:

1. она проста для понимания;
2. надежна и проста для сопровождения - правильное проектирование обеспечивает простоту расширения и модификации;

3. ускоряет цикл разработки – без многократного повторения компонентов уменьшается избыточность кода и снижает риск внесения ошибок при копировании.

Объектно-ориентированное программирование основано на:

1. Инкапсуляции;
2. Полиморфизме;
3. Наследовании.

Инкапсуляцией называется включение различных мелких элементов в более крупный объект.

Полиморфизм позволяет использовать одни и те же имена для похожих, но технически разных задач.

Наследование позволяет одному объекту приобретать свойства другого объекта, но это не копирование.

Исследования в области разработки с использованием принципов ООП не заканчиваются и по сей день. В статье [1] А. Околелов рассказывает о значимости знания ООП. А.В. Ересь [2] предлагает характеристики методов объектно-ориентированного программирования на PHP, а также продемонстрировано выполнение с помощью них поставленных задач. В статье Э.А. Усеинов [3] описано применение принципов объектно-ориентированного программирования на примере языка программирования Python, позволяющего легко реализовать кроссплатформенные приложения, а также представлены основные трудности при построении архитектуры объектно-ориентированных приложений.

Основным понятием в объектно-ориентированном программировании является класс. Классы образуют синтаксическую базу ООП, являясь своеобразным типом данных. Экземпляром класса является объект, который включает в себя свойства и методы, называющиеся членами класса. Проще говоря объекты - это всё то, что поддерживает инкапсуляцию.

Если класс примем за тип данных, то объект будет является переменной этого класса. Внутри класса может существовать несколько объектов с которыми можно одновременно работать. Члены класса могут находится как в открытом, так и в закрытом состоянии. Открытые данные могут быть доступны в других частях системы, что нельзя сказать о закрытых, которые открыты только внутри объекта.

Создадим первый класс, который будет выводить переменную (см. рис. 1).

```
class Name {
    public $name;
        public function getName()
    {
        $this->name = "Nick";
        echo $this->name;
    }
}
$new = new Name;
$new->getName();
```

Рисунок 1. Класс Name

В классе определяется публичная переменная name, которую можно изменить даже вне объекта. Далее создаётся метод getName, в котором идёт обращении к $name, изменение её значения на “Nick” и вывод. Вне класса, создаётся объект класса $new и обращение к методу (см. рис. 2).

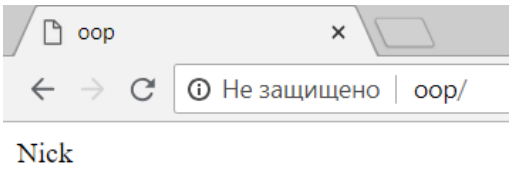

Рисунок 2. Результат создания класса

Как и говорилось раннее, в одном классе может быть множество методов, которые могут быть связаны между собой. Далее будет реализована простая авторизация с помощью принципов ООП. Для этого не понадобится БД. Каждый класс должен находится в отдельном файле. Для начала нужно создать класс (см. рис. 3).

```php
<?php
class authorization
{
    public $name;
    public $password;
    public $correctName = "Alexey";
    public $correctPassword = "2gqxch432";

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

    public function validate()
    {
        if ($this->correctName == $this->name & $this->correctPassword == $this->password)
        {
            echo 'yes';
        }else
        {
            echo 'no';
        }
    }

}
```

Рисунок 3. Класс авторизации

Класс authorization, в котором 4 переменных, 2 из которых уже определены. Далее идёт функция конструктор, которая принимает 2 параметра (то что будет вводить пользователь в поля на странице), которые записываются в переменные. Самое главное это сравнение данных, если данные совпадают, то выводится слово да, если нет, то нет. Код хорошо читаем и структурирован, теперь перейдем в главный html файл (см. рис. 4).

```php
<?php
require_once "models/authorization.php";
if ($_POST['action'] == "autho") {
    $obj = new authorization($_POST['login'],$_POST['password']);
    $obj->validate();
}
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>
<form method="POST">
    <input type="hidden" name="action" value="autho" >
<div class="container">
    <div class="row">
        <div class="fields">
            <input type="text" name="login" placeholder="Login">
        </div>
        <div class="fields">
            <input type="text" name="password" placeholder="Password">
        </div>
        <div class="fields">
            <input type="submit" placeholder="Авторизоваться">
        </div>
    </div>
</form>
</body>
</html>
```

Рисунок 4. Html документ с формами

Подключается класс, после идет проверка на отправку формы, если форма отправлена, то создаётся экземпляр класса и отправляются данные. Вызывается метод validate. В html коде 2 поля для ввода и кнопка (см. рис. 5).

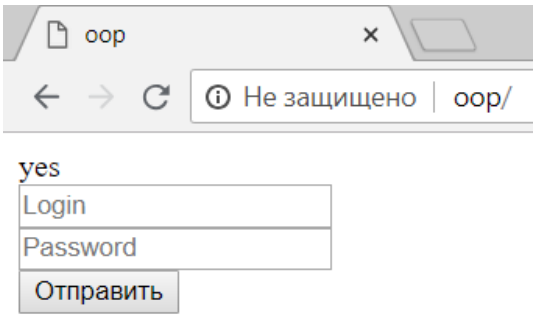

Рисунок 5. Результат попытки авторизоваться

Как и говорилось раннее, при правильном вводе данных выводиться слово да. В ООП огромное количество инструментов и методов для более упрощённого сопровождения и модернизации проекта. В данной статье использовался конструктор, принимающий данные с формы и метод для проверки. Всё строится на таких же методах и вызывается, когда это

необходимо. Если нужно было регистрировать пользователя, то был бы ещё один класс db с методом вывода статуса регистрации пользователя и т.д. Сейчас ООП необходим каждому современному программисту.

Библиографический список

1. Околелов А. ООП в php на практике // Системный администратор. 2011. № 12 (109). С. 78-81.
2. Ересь А.В. Методы объектно-ориентированного программирования на php // Постулат. 2018. № 8 (34). С. 39.
3. Усеинов Э.А. Объектно-ориентированное программирование в рамках дисциплины «Язык программирования python» // Ученые записки Крымского инженерно-педагогического университета. 2012. № 34. С. 157-160.
4. Объектно-ориентированное программирование // Основы программирования URL: http://kufas.ru/programming94.htm (дата обращения: 24.01.2019).