УДК 004.624

Обзор типичных проблем при объектно-ориентированном программировании на PHP

Ересь Артём Владимирович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье показаны типичные ошибки при проектировании в объектно-ориентированном программировании на PHP, подробно разобраны их причины и методы решения.
Ключевые слова: PHP, объектно-ориентированное программирование, проектирование

The review typical problems in case of object-oriented programming for PHP

Yeres Artem Vladimirovich
Sholom-Aleichem Priamursky State University
Student

Abstract
Typical errors in case of design in object-oriented programming for PHP are shown in this article, their reasons and methods of the decision explicitly are sorted.
Keywords: PHP, object-oriented programming, design

Ведение разработки проекта с использование объектно-ориентированного программирования достаточно сложный и трудоемкий процесс. Программисту необходимо хорошо продумать внутреннюю структуру, в частности создаваемые классы. А возможность разделения на модули является важной особенностью, так как это дает возможность с легкостью производить манипуляции с разрабатываемым проектом.

В связи с множеством важных аспектов в процессе разработки, со временем сложился ряд ошибок программиста, из-за которых созданное приложение в итоге может получится тяжело редактируемым и малофункциональным.

Целью данной работы является раскрытие основных проблем при разработке проекта в объектно-ориентированном программировании на PHP, выявление причин и способов решения.

Тема данной статьи является актуальной среди исследователей. Авторы Е.А. Гурьянова, Ю.В. Королькова рассмотрели существующие в сфере современного обучения проблемы процесса подготовки программистов для работы с объектно-ориентированным методом проектирования, предложили

способы их решения [1]. В работе Н.Р. Ахметова и А.А. Макарова рассмотрены важнейшие способы и основные направления использования объектно-ориентированных программных решений касательно автоматизации проектов [2]. И.Т. Степаненко и Е.В. Степаненко в своей статье затронули тему перехода от традиционного к объектно-ориентированному программированию, показали особенности каждого из них [3]. В интернет источнике представлены особенности объектно-ориентированного программирования на PHP, предложено руководство для начинающих пользователей [4]. Проведен разбор имеющихся ошибок программистов начального уровня знаний при работе с объектно-ориентированным методов в PHP [5].

Считается, что существует три основных вида ошибок при работе с объектно-ориентированным программированием. Подробно остановимся на каждом из них.

1. Потеря наследственной связи в классах

При создании структуры многие пользователи создают слишком много классов поэлементно для скриптов, и к тому же производят повторение частей. Поэтому использование кода с повторами считается неверным решение в работе.

Рассмотрим пример:

```
class IndexPage {
 protected $title;
 protected $db;

 public function __construct() {
 $this->db = new mysqli(HOST,USER,PASS,DB);
 }


 public function getContent () {
 $query = "Select * FROM articles";
 $data = $this->db->query($query);
 return $data;
 }

 public function render($data) {
 echo "<div>";
 echo $data;
 echo "</div>";
 }
}


class ArticlePage {
 protected $title;
 protected $db;

 public function __construct() {
 $this->db = new mysqli(HOST,USER,PASS,DB);
 }


 public function getContent ($id) {
 $query = "Select * FROM articles WHERE id = ".$id;
 $data = $this->db->query($query);
 return $data;
 }

 public function render($data) {
 echo "<div>";
 echo $data;
 echo "</div>";
 }
}
```

Рис. 1. Пример №1

Заметны два класса, практически одинаковые. И именно для этого существует наследственная связь, позволяющая провести распределение классов с точки зрения положения на уровнях. Обозначим способ решения проблемы, который значительно сокращен и упрощен.

```php
class Page {
 protected $title;
 protected $db;

 public function __construct() {
 $this->db = new mysqli(HOST,USER,PASS,DB);
 }

 public function render($data) {
 echo "<div>";
 echo $data;
 echo "</div>";
 }
}

class IndexPage extends Page {

 public function getContent () {
 $query = "Select * FROM articles";
 $data = $this->db->query($query);
 return $data;
 }

}

class ArticlePage extends Page {

 public function getContent ($id) {
 $query = "Select * FROM articles WHERE id = ".$id;
 $data = $this->db->query($query);
 return $data;
 }

}
```

Рис. 2. Решение проблемы с помощью наследуемости

2. Игнорирование интерфейса

Для описания рабочих функций методов программист должен брать разные классы со следующим кодом:

```php
class DbSave {
 public $db;

 public function __construct() {
 $this->db = new mysqli($host,$user,$pass,$db);
 }

 public function saveInDataBase($data) {
 if($data) {
 $query = "INSERT INTO comments(name,message)
 VALUES('".$data['name']."','".$data['message']."')";
 $result = $this->db->query($query);
 }
 }
}

class FileSave{

 public function saveInFile($data) {
 file_put_contents('messages.txt',$data);
 }
}
```

Рис. 3. Структура

Обратим внимание, что для работы с каждым классом необходимо прописывать дополнительные запросы, что является проблематичным для новичков в сфере программирования.

В этот момент помогает интерфейс определяющий метод доступности к содержимому, то есть работающие с ним классы переопределяются согласно запросу.

```php
interface ISave {
 public function save($data);
}

class DbSave implements ISave {
 public $db;

 public function __construct() {
 $this->db = new mysqli($host,$user,$pass,$db);
 }

 public function save($data) {
 if($data) {
 $query = "INSERT INTO post(name,messages)
 VALUES('".$data['name']."','".$data['messages']."')";
 $result = $this->db->query($query);
 }
 }
}

class FileSave implements ISave {

 public function save($data) {
 file_put_contents('file.txt',$data);
 }
}
```

Рис. 4. Использование интерфейса

3. Чрезмерная работа с public

Обычно считается простым и удобным напрямую из объектов отправлять запрос к их методам для объявления характеристик.

```php
<?php

class User {
 public $login;
}

$user = new User();
$user->login = 'ben';
echo $user->login;
```

Рис. 5. Обращение к объекту

Пример выше с легкостью справляется с этой задачей, однако при объектных манипуляциях, поправок в код не избежать. Это приносит сложности и в целом повышает длительность работы с проектом.

В примере ниже упростим метод:

```php
<?php

class User {
 private $login;

 public function setLogin($login) {
 $this->login = $login;
 }

 public function getLogin($login) {
 return $this->login;
 }
}

$user = new User();
$user->setLogin('ben');
echo $user->getLogin('ben');
```

Рис. 6. Исправление

Стоит отметить, что при работе с спецификаторами, доступ по прямой к некоторым частям класса невозможен. И поэтому для получения данных можно использовать set Login (), а также get Login (), что дает возможность не производить изменения на функциональном уровне.

Таким образом, в данной статье были рассмотрели типичные ошибки при проектировании в объектно-ориентированном программировании на PHP, подробно разобраны их причины и методы решения.

Библиографический список

1. Гурьянова Е.А., Королькова Ю.В. Проблемы обучения объектно-ориентированному программированию // Наука в современном мире. 2014. С. 39-42. URL: https://elibrary.ru/item.asp?id=22613342 (дата обращения: 26.08.2018)
2. Ахметов Н.Р., Макаров А.А. Объектно-ориентированные расширения в программировании систем автоматизации // Молодой ученый. 2015. № 13 (93). С. 96-100. URL: https://elibrary.ru/item.asp?id=23735662 (дата обращения: 26.08.2018)
3. Степаненко И.Т., Степаненко Е.В. Переход от традиционного к объектно-ориентированному программированию // Современные наукоемкие технологии. 2008. № 1. С. 26. URL: https://elibrary.ru/item.asp?id=9926657 (дата обращения: 26.08.2018)
4. Объектно-ориентированное программирование в PHP для начинающих URL: http://bourabai.ru/php/oop-php.htm (дата обращения: 26.08.2018)
5. Ошибки начинающий программистов при использовании объектно-ориентированного подхода URL: https://cyberleninka.ru/article/n/oshibki-nachinayuschih-programmistov-pri-ispolzovanii-obektno-orientirovannogo-podhoda (дата обращения: 26.08.2018)