Автор: Казимиров К.О.
20.04.2017 10:11 -

О ПРИМЕНЕНИИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА ПРИ РАЗРАБОТКЕ КОМПОНЕНТА «ПРЕДСТАВЛЕНИЕ» (VIEW) ДЛЯ СИСТЕМЫ УЧЁТА НАУЧНОЙ ДЕЯТЕЛЬНОСТИ ПРЕПОДАВАТЕЛЕЙ

Казимиров К.О., студент

г.Бирск, Бирский филиал БашГУ

Я занимаюсь разработкой информационной системы (ИС) учёта научной деятельности преподавателей. ИС представляет собой оболочку для работы с базой данной написанною на языке PHP с SQL. Сама база данных представляет собой несколько таблиц хранящих труды преподавателей высшего учебного заведения. Здесь мы не будем углубляться в структуру базы данных, так как речь идёт об оптимизации кода путём использования объектно-ориентированное программирования(ООП). Стоит лишь знать, что в таблицах используются поля трёх типов: «текст», «число», «выбор».

Для оптимизации количества кода, можно даже сказать его генерации мы воспользуемся следующими понятиями из ООП:

· Абстракция – подразумевает выделение главной информации, путём исключения незначительной.

· Инкапсуляция – позволяет произвести сокрытие данных от внешнего вмешательства или неправильного использования.

· Наследование – позволяет описать новый класс на основе уже существующего с

заимствованием его функциональностью. Класс от которого происходит наследование называют базовым или родительским классом. Новый класс – потомком или наследником.

· Класс – тип данных, являющийся моделью информационной сущности с внешними и внутренними интерфейсами. Данные класса хранятся в свойствах, а работу над ними выполняют методы.

· Полиморфизм – использование объектов с одинаковыми интерфейсами без информации о типе объекта.

· Объект – экземпляр класса.

Для создания класса на PHPдостаточно написать два слова и две фигурные скобки. Вначале идёт ключевое слово **class**, затем название класса, открывающая и закрывающая фигурная скобка. Пример «
class
NameClass
{}
». В фигурных скобках заключается тело класса, в нем описываются поля и методы для работы с ними. Для работы с членами класса используется оператор «->», а для использования члена класса внутри класса применяется ключевое слово «$
this
» с оператором «->», пример «
$
this
->
»
[1]
.

Для последующего использования класса нам потребуется создать его объект, для этого используется ключевое слово **new**. Пример «**newNameClass**». Но для использования некоторых методов и свойств, нам не требуется создавать отдельно объекты класса, такие поля и методы имеют ключевое слово
static
при описании.

Так-же можно использовать модификаторы доступа **public, private, protected**[1]**.**

· **Public** – модификатор доступа, означает что свойство или метод открыт для всех объектов, которые их используют. Мы сможем обращаться к ним напрямую из любого места.

· **Protected** – данный элемент будет доступен внутри самого объекта и его наследниках.

· **Private**- означает, что данный элемент объекта может быть использован только в самом объекте и нигде больше, даже в самих объектах.

Использование модификаторов доступа позволяет нам воспользоваться понятием **инкапсуляция**
.

Для наследования класса мы должны воспользоваться следующей структурой кода: **class**
класс_наследника
extends
класс_родителя{}
. После наследования мы можем использовать методы и поля класса родителя, если

это позволяют модификаторы доступа
[2]
.

До использования ООП для каждого типа научной деятельности нам приходилось создавать несколько функций для работы с ними, например, такие как показ таблицы, формы добавления. Сейчас при использовании ООП мы создали один класс родитель, с названием «**nir**» при каждом добавлении нового типа научной работы, мы создаём наследника nir, описываем в нём свойства, а методы наследуем от родителя.

Приведём пример для демонстрации всех плюсов ООП.

Создадим class nir со следующими свойствами «$fields_name, $fields_description, $fields_type, $table_name». Заметим, что все свойства имеют модификатор
protected
. Опишем в нём несколько методов:

```
protected function create_table_head(){

echo '<table class="table table-striped table-bordered table-hover table-condensed"
data-table-name="'.$this->name.'">';
```

```
echo  '<tr><td>Номер</td>';

for($i=0;$i<count($this->fields_name);$i++)echo  '<td>'.$this->fields_description[$i].'</td>';

echo  '</tr>';

}

Public function show_table(){

$this->create_table_head();

$req  = "Select * From ".$this->name;

$result=mysql_query($req);

$i;

while  ($row=mysql_fetch_array($result)){

    echo '<tr  name="'.$row[0].'"><td>'.++$i.'</td>';

    echo  '<td>'.$row[1].'</td><td>'.$row[2].'</td><td>'.$row[3].'</td>';
```

```
echo'<td>'.$array[$row[4]].'</td><td>'.$row[5].'</td><td>Удалить</td></tr>';

}

}
```

При вызове функции show_table будет показана таблица со всем содержимом. До внедрения принципов ООП, нам бы пришлось дублировать этот код (с некими малыми изменениями) для каждого нового типа работы, теперь же всё очень просто – создаём наследника класса, и задаём ему свойства.

```
class  Articles extends nir{

public  function __construct()

   {

      $this->fields_name =  array("names","publishing","pages","type","year");

         $this->fields_description  = array("Название","Издательскиеданные","Количество
```

```
страниц","articles","Год");

        $this->fields_type =  array("text","text","digit","select","digit");

        $this->name =  "articles";

        $this->url =  "/add.php";

}

}
```

Готово теперь при вызове следующего кода «**$obj = newArticles(); $obj->show_table();** » будет показана, таблица articles

со всеми полями из массива $fields_name.

На основе анализа кода без ООП можно сделать вывод, что в среднем на один тип материала, для показа таблицы уходило примерно 900 символов. Новый метод оценивается в 750 символов, что даёт уже прирост в 150 символов, но это ещё не всё при добавлении новых типов материалов раньше требовалось создать новый метод. Из всего этого можно сделать вывод, что до использования ООП нам приходилось тратить

на создание метода показа содержимого таблицы 900*n, где n – количество различных типов научных работ. После внедрения новой методологии мы получаем фиксированное количество символов 150, не зависящих от количества типов.

Вывод

Использование методологии объектно-ориентированного программирования, позволяет нам сэкономить большое количество времени на написание кода, сделать код более чистым и уменьшить время на добавление новых функций.

Литература

1. 1.Зандстра, Мэт PHP. Объекты, шаблоны и методики программирования/ М. Зандстра. –М.: ООО ”Вильямс”, 2015. -576с

2. 2.Гамма, Э. Приёмы объектно-ориентированного проектирования. Паттерны проектирования/ Э. Гамма [и д.р.] –СПб.:Питер, 2010. -366с