УДК 004

Реализация паттерна проектирования Factory на PHP

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

Аннотация
В статье рассматривается один из самых популярных паттернов проектирования Factory на языке программирования PHP. В качестве примера реализована задача с выбором сложности в веб-игре.
Ключевые слова: Factory, проектирование, паттерн.

Implementation design pattern Factory on PHP

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article implements one of the most popular Factory design patterns in the PHP programming language. As an example, implemented the task with the choice of complexity in the web game.
Keywords: Factory, design, pattern.

Фабрика (Factory) - один из наиболее часто применяемых шаблонов проектирования в программировании, обычно его используют в случае, когда во время исполнения программы необходимо выбрать один из взаимозаменяемых классов.

Шаблон проектирования Factory направлен на решение проблемы сильно связанных классов в больших приложениях. Она позволяет создавать объекты без вызова new.

Существует несколько способов реализации этого шаблона:

1. Простая фабрика (Simple Factory).
2. Абстрактная фабрика (Abstract Factory).
3. Фабричный метод (Factory Method).

В этой статье рассматривается последний из перечисленных способов, так как он более близок к официальному определению шаблона и чаще находит применение в практическом программировании.

Шаблон Factory стоит использовать тогда, когда необходимо скрыть логику создания сложного объекта или уменьшить сильные связи между классами приложения.

Далее рассмотрим практические примеры использования шаблона проектирования Factory. (см. рис. 1).

На сегодняшний день исследования в области применения паттернов программирования актуальны. В статье П.П. Кейно, Ф.Ф. Ярмухаметов [1] описывается архитектура класса элемента и его дочерних классов, дается детальное обоснование использования паттерна «абстрактная фабрика». С.М. Гардейчик, А.И. Шербаф [2] приводят краткое описание PHP-фреймворка Laravel, реализующего шаблон MVC. В статье О.Н. Симонова [3] рассказывается о шаблонах проектирования MVC как эффективное средство построения архитектуры программной системы. Р.И. Ибраимов [4] создаёт тесты для spring mvc контроллеров. А. Майоров [5] представлен один из вариантов реализации Singleton. В статье П.П. Олейник [6] рассмотрена реализация шаблона проектирования Singleton, представляющая собой атрибут, который можно использовать в языках платформы Net Framework. Проведен анализ существующих реализаций данного паттерна на языке C#, выявлены его достоинства и недостатки. В.К. Хорошхин описывает использование паттерна Singleton в языках программирования C++, C#, Java.

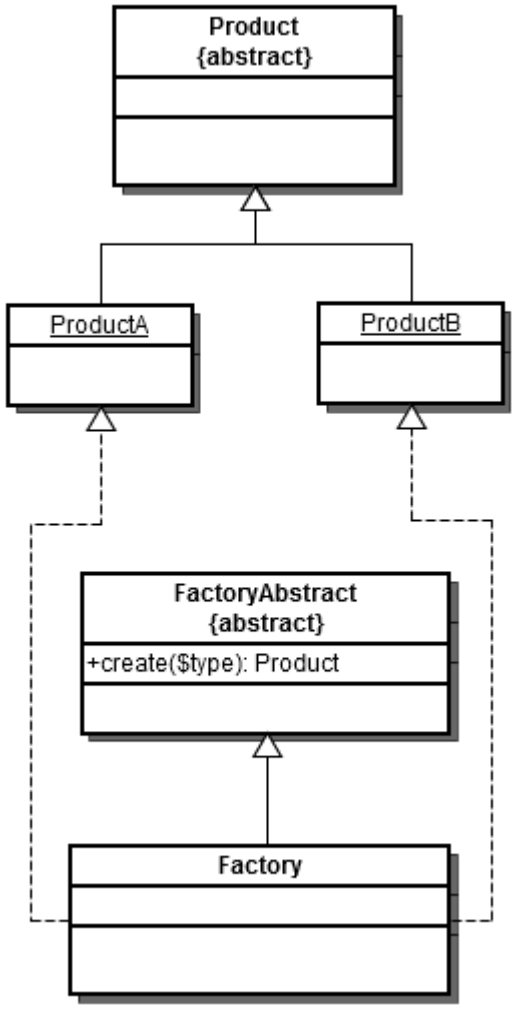

Рисунок 1. Модель Factory

В данной статье будет реализован паттерн на примере выбора уровня сложности в игре. Представим, что перед программистом стоит задача

реализовать веб-игру, в которой, имеются 3 уровня сложности. Уровни сложности отличаются от количества элементов, которое будет сгенерировано на поле битвы. То есть элементы во всех 3 случаях будут абсолютно одинаковые, что нельзя сказать об их количестве. Когда этих элементов мало, то можно воспользоваться стандартным switch и не усложнять игру, но когда этих элементов много, необходимо структурировать код для дальнейшего сопровождения. Рассмотрим главный файл (см. рис. 2).

```php
<?php
require_once 'factory.php';
if ($_POST['action'] == 'choice')
{
    $factory = factory::switchLevel($_POST['level']);
    echo '<br>';
    var_dump($factory);
}
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>factory</title>
</head>
<body>
<div class="container">
    <div class="row">
        <form method="POST">
            <input type="hidden" value="choice" name="action">
            <input type="submit" value="Лёгкий" name="level">
            <input type="submit" value="Средний" name="level">
            <input type="submit" value="Сложный" name="level">
        </form>
    </div>
</div>
</body>
</html>
```

Рисунок 2. Файл index

В главном файле есть 3 кнопки с выбором уровня сложности. Если сложность выбрана то, обращаемся к методу switchLevel и передаём переменную с выбранным уровнем сложности. Перейдём к классу (см. рис. 3,4).

```php
<?php
abstract class level
{
    public $level;
    public function __construct($level)
    {
        echo $this->level = $level;
    }
}
class easy extends level
{
    public function __construct($level)
    {
        parent::__construct($level);
    }
}
class medium extends level
{
    public function __construct($level)
    {
        parent::__construct($level);
    }
}
class heavy extends level
{
    public function __construct($level)
    {
        parent::__construct($level);
    }
}
```

Рисунок 3. Классы уровней сложности

```php
class factory
{
    public static function switchLevel($level){
        switch ($level)
        {
            case 'Лёгкий':
                return new easy( level: 'easy');
                break;
            case 'Средний':
                return new medium( level: 'medium');
                break;
            case 'Сложный':
                return new heavy( level: 'heavy');
                break;
        }
    }
}
```

Рисунок 4. Класс Factory

Сначала идёт абстрактный класс. В нём просто выводится переданная переменная. Далее 3 класса, каждый из которых отвечает за свой уровень сложности, внутри может быть вставлена генерация элементов игры. Все классы наследуют level. После создаётся главный класс Factory с методом switchLevel, в котором идёт выбор вызываемого класса. Переменная level передаётся с index файла через вызов метода. Далее следует результат выбора сложного уровня (см. рис. 5).

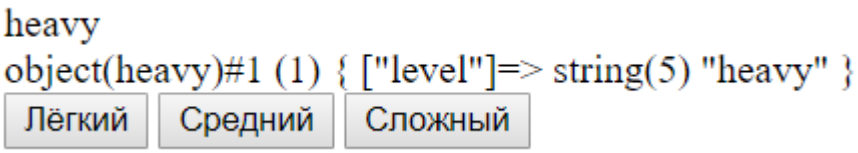

Рисунок 5. Результат

При выборе сложного уровня сначала вызвалась переменная, а после для наглядности массив со всеми типами и составляющими. Был выбран объект heavy в котором level = heavy. Для такого шаблона проектирования задача слишком простая, этот паттерн используется в огромных системах, где выборки могут достигать сотни и выше. Программисты часто используют данный инструмент для решения различного рода задач.

Рассмотренный паттерн решает широкий спектр и позволяет сохранить веб-проект в приемлемом для дальнейшей работы виде, кроме того он достаточно прост в использовании.

Библиографический список

1. Кейно П.П., Ярмухаметов Ф.Ф. Использование паттерна «абстрактная фабрика» в реализации модуля валидации и преобразования данных интерпретатора // Прикладная информатика. 2017. Т. 12. № 1 (67). С. 95-101.
2. Гардейчик С.М., Шербаф А.И. PHP-фреймворк laravel с использованием архитектурной модели MVC // Перспективные направления развития отечественных информационных технологий Материалы III межрегиональной научно-практической конференции. Научный редактор Б.В. Соколов. 2017. С. 133-135.
3. Симонова О.Н. Шаблон проектирования mvc как эффективное средство построения архитектуры программной системы // Студенческий научный форум -2014 VI Международная студенческая электронная научная конференция: Электронное издание. 2014.
4. Ибраимов Р.И., Джемалетдинов А.Б., Шевченко А.А. Spring boot: создание тестов для spring mvc контроллеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2018. № 4 (18). С. 104-111.
5. Майоров А. Варианты реализации паттерна singleton // Системный

администратор. 2013. № 1-2 (122-123). С. 92-95.
6. Олейник П.П. Декларативный подход к реализации шаблона проектирования Singleton // Информационные технологии и вычислительные системы. 2006. № 4. С. 45-51.
7. Хорошхин В.К. Использование паттерна singleton в языках программирования c++, c#, java // Труды Университета. 2011. № 1. С. 82-84.
8. Шаблоны проектирования в PHP : Фабрика // devacademy URL: http://devacademy.ru/posts/shablonyi-proektirovaniya-v-php-fabrika/ (дата обращения: 25.01.2019).