УДК 004

Реализация паттерна проектирования Observer на PHP

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

Аннотация
В статье рассматривается один из самых популярных паттернов проектирования Observer на языке программирования PHP. В качестве примера реализована задача отслеживания подписчиков журнала.
Ключевые слова: Observer, проектирование, паттерн.

Implementation design pattern Observer on PHP

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article implements one of the most popular Observer design patterns in the PHP programming language. As an example, the task of tracking journal subscribers is implemented.
Keywords: Observer, design, pattern.

Observer (Наблюдатель) - является поведенческим шаблоном проектирования. Является достаточно популярным шаблоном проектирования, но, при этом, очень прост в реализации. Данный шаблон предполагает зависимость между объектами "один ко многим" так, что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются автоматически. Таким образом, в шаблоне наблюдаются две роли: субъект и слушатель. Шаблон "Наблюдатель" определяется следующими свойствами:

- существует, как минимум, один субъект, рассылающий сообщения;
- имеется не менее одного получателя сообщений, причём их количество и состав могут изменяться во время работы приложения;
- нет надобности очень сильно связывать взаимодействующие объекты, что полезно для повторного использования.

Шаблон подходит для любого сценария, в котором требуется использование push-уведомлений. Очень часто его можно заметить в системах пользовательского интерфейса.

Типовая UML-диаграмма выглядит следующим образом (см. рис. 1).

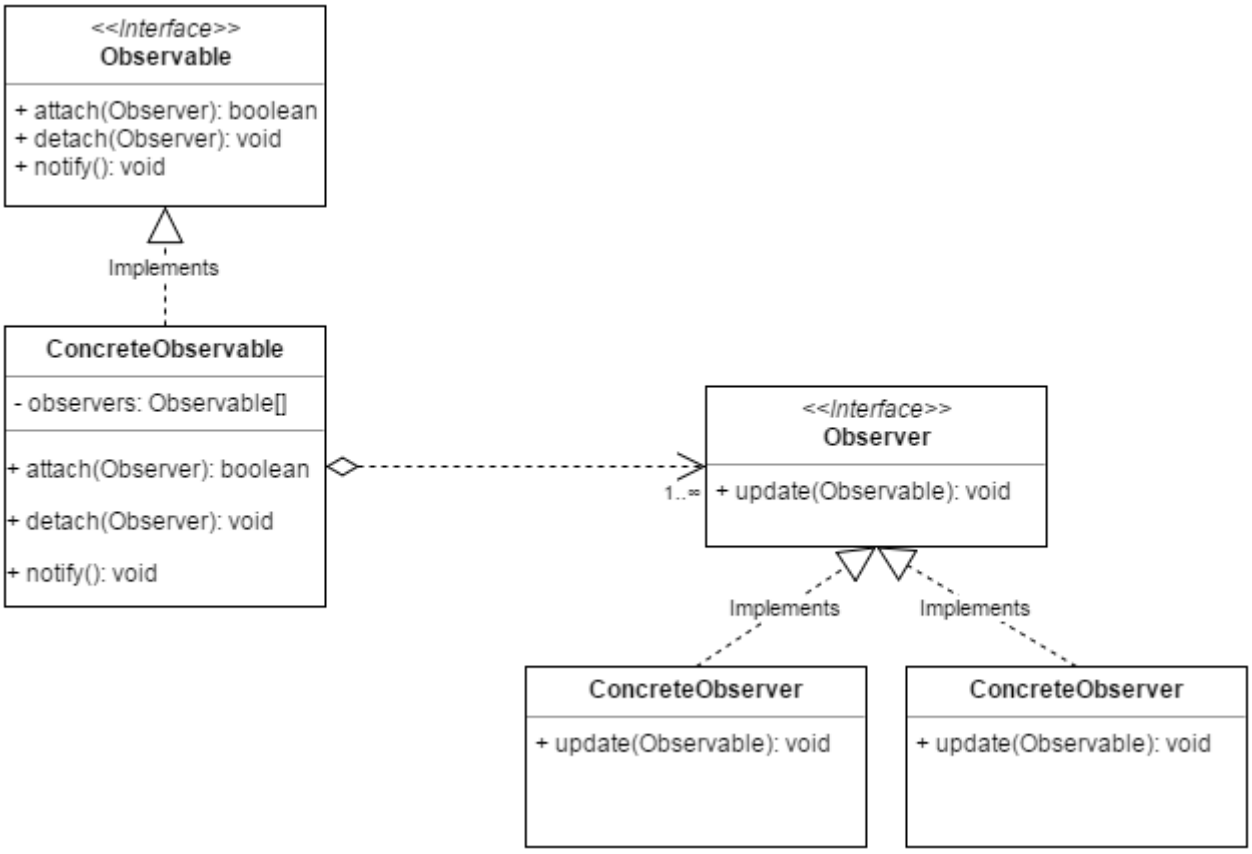

Рисунок 1. UML-диаграмма

На сегодняшний день исследования в области применения паттернов программирования актуальны. В статье В.Г. Машкова, Н.Е. Сусоева, А.Г. Горбунова [1] проводится анализ лучших решений объектно-ориентированного программирования. Одним из выбранных решений является использование шаблона проектирования Observer. А.Н. Моисеев [2] предоставляет решение задачи по проектированию с повышенной степенью повторного использования контроллера. В решении рассматривается и применяется паттерн Observer. В работе Е.А. Козырева [3] описывается способ решения сложных задач, требующих отслеживания важных манипуляций, которые могут нарушить целостность данных или работоспособность системы. А. Пахомов [4] в статье использует паттерн Observer для разработки мобильного приложения.

В данной статье будет реализован паттерн на примере подписчиков журнала. Задача заключается в том, чтобы реализовать систему отслеживания подписчиков журнала. Данный шаблон будет решать эту задачу. (см. рис. 2,3,4).

```php
<?php
interface SubjectInterface
{
    public function attach($observer);
    public function detach($observer);
    public function notify();
}
interface ObjectInterface
{
    public function update($subject);
}
class Journal implements SubjectInterface
{
    private $observers = [];
    private $name;
    public function __construct($name)
    {
        $this->name = $name;
    }
    public function attach($observer)
    {
        $this->observers[] = $observer;
    }
    public function getName()
    {
        return $this->name;
    }
    public function detach($observer)
    {
        foreach ($this->observers as $key => $val) {
            if ($val === $observer) {
                unset($this->observers[$key]);
                return;
            }
        }
    }
```

Рисунок 2. Реализация системы подписок

```php
class Subscribers implements ObjectInterface
{
    private $name;
    public function getName()
    {
        return $this->name;
    }
    public function __construct($name)
    {
        $this->name = $name;
    }
    public function update($nameJournal)
    {
        echo ($this->getName() . ' подписан(а) на ' .  $nameJournal->getName());
        echo '<br>';
    }
}
```

Рисунок 3. Реализация системы подписок

```php
$nameJournal1 = new Journal( name: 'Maxim');
$subscribers1= new Subscribers( name: 'Василий');
$subscribers2= new Subscribers( name: 'Мария');
$subscribers3= new Subscribers( name: 'Владимир');
$subscribers4= new Subscribers( name: 'Алексей');
$subscribers5= new Subscribers( name: 'Георгий');
$nameJournal1->attach($subscribers1);
$nameJournal1->attach($subscribers2);
$nameJournal1->notify();
echo '<br>';
$nameJournal1->attach($subscribers3);
$nameJournal1->notify();
echo '<br>';
$nameJournal1->detach($subscribers3);
$nameJournal1->attach($subscribers4);
$nameJournal1->attach($subscribers5);
$nameJournal1->notify();
```

Рисунок 4. Реализация системы подписок

Сначала нужно создать 2 интерфейса. В данном случае объектом является человек, который хочет подписаться на журнал, а субъектом будет сам журнал. Далее идёт класс, который наследует функции от интерфейса субъекта. Объявляется приватный массив, который будет хранить подписчиков, а другой имя журнала. В конструктор, который инициализируется при создании экземпляра, записывается переменная с названием журнала. Метод attach принимает имя подписчика и записывает его в массив к другим, а getName просто возвращает имя журнала. Далее при вызове метода detach идёт поиск по массиву подписчиков и если есть

совпадение, то он удаляется. Последний в классе метод notify, который отвечает за уведомления о подписанных на журнал пользователей.

Класс подписчиков наследует интерфейс объекта. Объявляется приватная переменная, которая хранит в себе имя подписчика. Конструктор так же, как и субъект, просто перезаписывает имя нового подписчика. Метод update выводит информацию о всех существующих журналах и подписчиков на него.

Далее идёт заполнение подписчиков в журнал. Создаётся экземпляр журнала и передаётся его имя. После описываются 5 подписчиков, которые пока просто зашли в журнал, но не подписаны на него. При вызове метода attach из простых пользователей, Василий и Мария стали подписчиками. Метод detach удаляет подписчика, если таковой имеется. Результат работы системы подписок журнала (см. рис. 5).

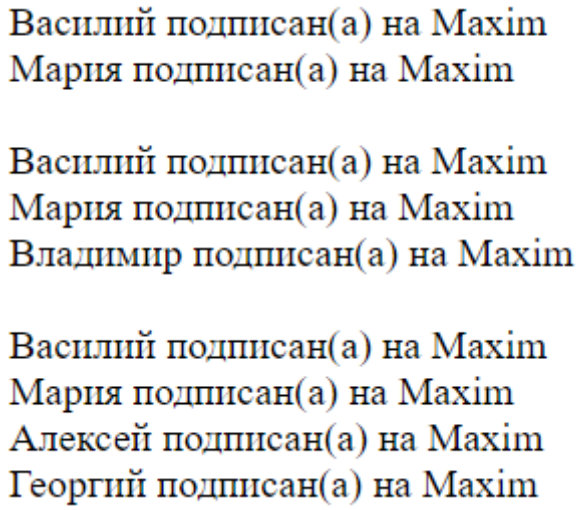

Рисунок 5. Реализация системы подписок

Сначала было подписано 2 пользователя, после чего был вызван метод уведомления о них, после подписался ещё один подписчик, а в 3 случае Владимир отписался, но был заменён сразу двумя новыми.

Рассмотренный паттерн может быть использован для решения задач, где присутствует рассылка уведомлений о появлении обновления или нового материала.

Библиографический список

1. Машков В.Г., Сусоев Н.Е., Горбунов А.Г. Применение шаблона observer при проектировании систем автоматизированного освоения // Инновационные технологии в образовательном процессе Материалы XIX Всероссийской научно-практической конференции. 2017. С. 163-167.
2. Моисеев А.Н. Контроллер интерфейса пользователя с повышенной степенью повторной используемости // Вестник Томского государственного университета. 2006. № 293. С. 156-157.

3. Козырев Е.А., Бурданова Е.В. Реализация шаблона проектирования "наблюдатель" на php // Universum: технические науки. 2018. № 6 (51). С. 20-22.
4. Пахомов А. Пишем мобильные приложения быстрее. паттерн наблюдатель и реактивное программирование // Системный администратор. 2016. № 5 (162). С. 44-47.