УДК 004.65:004.9
DOI: 10.31866/2617-796X.4.2.2021.247488

Ткаченко Ольга,
кандидат фізико-математичних наук,
доцент кафедри інформаційних технологій та дизайну,
Державний університет інфраструктури та технологій,
Київ, Україна
oitkachen@gmail.com
https://orcid.org/0000-0003-1800-618X

Русаков Микита,
магістрант, кафедра інформаційних технологій та дизайну,
Державний університет інфраструктури та технологій,
Київ, Україна
zeeynnik@gmail.com
https://orcid.org/0000-0002-0248-2520

ДЕЯКІ АСПЕКТИ АВТОМАТИЗОВАНОГО ВПРОВАДЖЕННЯ ЗАЛЕЖНОСТЕЙ У МОВІ PHP

Метою статті є дослідження, аналіз і розгляд загальних проблем і перспектив використання принципу впровадження залежностей під час розробки вебзастосунків мовою PHP.

Методами дослідження є методи семантичного аналізу основних понять цієї предметної сфери (вебтехнології та вебзастосунки). У статті розглянуто відомі підходи до інверсії управління на основі використання шаблону впровадження залежностей.

Новизною проведеного дослідження є розв'язання проблем інверсії управління на основі шаблону під час проєктування вебзастосунків.

Висновки. Проаналізовано наявні проблеми та перспективи застосування принципу впровадження залежностей у вебзастосунках, що мають програмний код, написаний мовою PHP. Враховуючи результати проведеного аналізу, розроблено систему автоматизації впровадження залежностей, яка має важливе значення для розв'язання проблем підвищення ефективності процесів розробки вебзастосунків мовою PHP.

Ключові слова: вебзастосунок; проєктування вебзастосунків; упровадження залежностей; PHP; ECMAScript; JavaScript.

Вступ. Розвиток вебтехнологій зумовлює розв'язання складних і комплексних проблем як на сервері, так і на фронт-енді. У першому випадку для реалізації є багато рішень (наприклад, JavaScript). Крім того почали розвиватися мови програмування для розробки вебзастосунків, з'явилися нові специфікації ECMAScript (Kulkarni).

У статті пропонуємо розглянути реалізацію принципу програмування Інверсія управління (Inversion of Control), який застосовує в процесі такий шаблон проєктування, як Впровадження залежностей (Dependency Injection). Під час реалізації такого підходу до проєктування вебзастосунків виникає багато проблем, пов'язаних як із сервером, так і з клієнтом (проблеми server, front-end, back-end).

Тому актуальність проблеми застосування принципу інверсії та впровадження залежностей не викликає сумнівів. Для розкриття вказаної вище проблеми введемо декілька визначень основних понять, які пропонуємо до розгляду.

Інверсія управління – принцип програмування, що передбачає передачу керуванням потоком (викликом команд) конкретній реалізації цього підходу (фреймворком), що є зовнішньою щодо коду вебзастосунку розробника (Kukurba, 2019; Инверсия и внедрение зависимостей, 2021).

Уживання поняття фреймворку чи каркаса обумовлено тим, що розробник використовує його під час реєстрації частини свого коду (методів, класів, модулів, процедур тощо), і цей фреймворк/каркас може визначати, коли викликати той чи той компонент програмного коду вебзастосунку.

Залежність (Dependency) – об'єкт, або будь-який програмний unit, що використовується в клієнті – іншому об'єкті / програмному unit (Kukurba, 2019; Инверсия и внедрение зависимостей, 2021). Часто такий об'єкт використовується в різних місцях IT-проєкту (наприклад, у вебзастосунку чи вебзастосуванні).

Упровадження залежностей – один з підходів для реалізації інверсії управління через композицію залежностей у тому чи тому залежному unit (Dependency Injection (Внедрение зависимостей); Русаков, 2017). Є декілька варіантів упровадження залежностей, зокрема через конструктор (у такому разі залежність передається як окремий параметр у відповідному методі конструктора), сеттер, інтерфейс.

Результати дослідження. Упровадження залежностей – шаблон проєктування програмного забезпечення на основі використання парадигми об'єктно-орієнтованого програмування (Что такое внедрение зависимостей и как это использовать в разработке?; Dependency Injection (Внедрение зависимостей)). Такий шаблон передбачає надання зовнішньої залежності програмному компоненту, використовуючи інверсію управління для отримання залежностей.

Використання цього шаблону програмування (це стосується й мови програмування PHP) можна означити за допомогою, зокрема, таких сутностей (Русаков, 2017; PHP Инъекция Зависимостей):

– клас, який упроваджує сервіси;
– створюваний об'єкт;
– сервіси, які впроваджуються.

У створюваному об'єкті під час його проєктування слід уникати створення додаткових об'єктів напряму, натомість необхідно передати цю властивість сторонньому сервісу, який створюватиме потрібні для роботи створюваного об'єкта сервіси та передаватиме його в конструктор нового класу.

Отже, під час створення проєкту за допомогою впровадження залежності слід передбачити впровадження всіх об'єктів сервісів, які необхідні для створення нового класу (PHP Инъекция Зависимостей).

Наприклад, маємо клас A (рис. 1). Клас A має метод sayHi(), завдання якого вивести рядок «Hello World» (цей приклад часто наводиться під час опису багатьох мов об'єктно-орієнтованого програмування). Такий клас не має задекларованих методів будування необхідного рядка, але використовує сервіси B та C (рис. 2), які мають необхідні для виводу методи. Якщо викликати вищеописаний метод класу

А, то під час роботи буде отримана фатальна помилка, яка повідомлятиме, що необхідні сервіси не були включені, отже, слід продовжити проєктувати систему.

```php
class A
{
    private B $b;
    private C $c;

    public function __construct(
        B $b,
        C $c
    ) {
        $this->b = $b;
        $this->c = $c;
    }

    public function sayHi()
    {
        echo $this->b->hello() . ' ' . $this->c->world();
    }
}
```

Рис. 1. Код для опису класу A

Для реалізації системи впровадження залежностей треба реалізувати сервіс- упроваджувач. Назвемо цей сервіс (для прикладу, що розглядаємо) ObjectManager. Для цих класів треба ввести декілька важливих зауважень:

– ObjectManager повинен мати два основних методи – get() та create();

– усі методи цього сервісу мають бути доступні без використання об'єктів, отже, виклик цих методів має бути статичним;

– обидва методи мають повертати готові об'єкти класів;

– після створення сервісу, виконавши необхідну модифікацію, буде отримано результат, показаний на рис. 3. Сервіс має змінну $objectPull типу масив. Ця змінна необхідна для того, щоб не створювалися нові об'єкти одного й того ж самого класу.

```php
class B
{
    public function hello()
    {
        return 'Hello';
    }
}

class C
{

    public function world()
    {
        return 'World';
    }
}
```

Рис. 2. Код для опису класів B і C

```php
class ObjectManager
{
    /**
     * @var array
     */
    private static array $objectPull = [];

    /**
     * @param string $classname
     * @return object
     */
    public static function get(string $classname): object
    {
        if (!self::$objectPull[$classname]) {
            self::$objectPull[$classname] = self::create($classname);
        }

        return self::$objectPull[$classname];
    }

    /**
     * @param string $classname
     * @return object
     */
    public static function create(string $classname): object
    {
        return new $classname();
    }
}
```

Рис. 3. Простий сервіс створення об’єктів

Кожен новий об’єкт після створення заноситься до цієї змінної як елемент масиву і за нової спроби створити новий об’єкт спочатку перевіряється його наявність у системі. Уже зараз можливо побачити один з недоліків такого шаблону – якщо використовується велика кількість різних класів, наприклад більше однієї тисячі, то система буде опрацьовувати всю інформацію повільніше, але все ж таки швидше, ніж створення нових об’єктів без збереження вже наявних.

Оскільки об’єктно-орієнтоване програмування в PHP реалізовано за допомогою простору імен, то змінна $className, яку можна побачити у створених методах, під час використання буде містити рядок виду «Path\To\Class» і допомагатиме створювати нові об’єкти.

На сьогодні проєктування системи Менеджер об’єктів може створювати лише об’єкти тих класів, що не мають залежностей, тому необхідно допрацювати його методи. Для подальшого проєктування сервісу зі створення об’єктів класів буде використовуватися насамперед шаблон програмування «Однак» (англ. Singleton). Під час проєктування будь-якої системи, у тому числі й вебзастосунку, розробле-

ного засобами мови PHP, за шаблоном «Однак» слід узяти до уваги, що вся робота із системою буде вестися за допомогою одного об'єкта класу, тобто за допомогою одного об'єкта будуть створюватися та працювати всі інші підсистеми.

Для перетворення класу ObjectManager в одинака слід додати до нього ще один метод – getInstance(). Цей метод є статичним, що сприяє його виклику без створення об'єкта; у результаті отримаємо одну змінну, яка охоплюватиме всі об'єкти, що були створені під час роботи в системі (рис. 4).

```php
    /**
     * @return static
     */
    public static function getInstance(): self
    {
        if (!self::$instance instanceof self) {
            self::$instance = new self();
        }

        return self::$instance;
    }
```

Рис. 4. Метод отримання об'єкта-одинака

```php
<?php
declare(strict_types=1);

namespace Framework;

class ConstructorDefinitionReader implements DefinitionReaderInterface
{
    /**
     * @param string $classname
     * @return array
     * @throws \ReflectionException
     */
    public function read(string $classname): array
    {
        $result = [];
        $reflection = new \ReflectionClass($classname);
        $constructor = $reflection->getConstructor();

        if ($constructor) {
            foreach ($constructor->getParameters() as $constructorParameter) {
                $result[$constructorParameter->getName()] = $constructorParameter->getClass()->getName();
            }
        }

        return $result;
    }
}
```

Рис. 5. Зчитувач конструктора

Наступним кроком у вдосконаленні сервісу зі створення об'єктів є проєктування підсистеми, яка б дала змогу отримувати зі створюваного об'єкта класу його залежності, а також їх створення. Для цього завдання треба створити так званий підсервіс, який назвемо ConstructorDefinitionReader – зчитувач залежностей з конструктора класу створюваного об'єкта (рис. 5).

Для того щоб отримати аргументи конструктора класу створюваного об'єкта використовується стандартна бібліотека мови PHP «Reflection».

Ця бібліотека дає змогу отримати всю інформацію про клас, обходячи інкапсуляцію. Хоча вищезгадана бібліотека охоплює значну кількість функціоналу, для створення зчитувача буде використана лише та її частина, яка дає змогу працювати з аргументами конструктора.

Для отримання параметрів у зчитувачі був створений метод read(), який у результаті роботи повертатиме масив аргументів, та їхні імена класів.

Після створення зчитувача треба доповнити клас ObjectManager, а саме додати в його конструктор декларацію зчитувача та впровадити цей зчитувач під час створення його об'єкта (рис. 6, рис. 7).

```php
/**
 * @param ConstructorDefinitionReader $constructorDefinitionReader
 */
public function __construct(ConstructorDefinitionReader $constructorDefinitionReader)
{
    $this->constructorDefinitionReader = $constructorDefinitionReader;
}
```

Рис. 6. Удосконалення конструктора

Такі вдосконалення дадуть змогу уникнути помилок під час першого створення об'єкта-одинака.

```php
/**
 * @return static
 */
public static function getInstance(): self
{
    if (!self::$instance instanceof self) {
        $definitionReader = new ConstructorDefinitionReader();
        self::$instance = new self($definitionReader);
    }

    return self::$instance;
}
```

Рис. 7. Удосконалення методу getInstance()

Тепер слід доробити метод create() менеджера об'єктів, що проєктується. Який на вигляд цей метод, можна побачити на рис. 3, після деяких удосконалень метод матиме такий вигляд (рис. 8).

Код, наведений на рис. 8, охоплюючи номери рядків, описує вдосконалений метод, який стає надто складним для швидкого розуміння, і тому виникає необхідність пояснення кожного рядка. На 70 рядку зчитувач конструктора перевіряє аргументи конструктора створюваного класу та повертає масив його значень. На наступному рядку створюється змінна, що являє собою стек. Її використання описано нижче.

```
63      /**
64       * @param string $classname
65       * @return object
66       * @throws \ReflectionException
67       */
68      public function create(string $classname): object
69      {
70          $parameters = $this->constructorDefinitionReader->read($classname);
71          $stack = [];
72          $this->creationStack[$classname] = true;
73
74          foreach ($parameters as $argumentName => $type) {
75              if (isset($this->creationStack[$type])) {
76                  throw new \LogicException( message: 'Cyclomatic dependency!');
77              }
78
79              $stack[] = $this->get($type);
80          }
81
82          unset($this->creationStack[$classname]);
83
84          return new $classname(...$stack);
85      }
86  }
```

Рис. 8. Удосконалення методу створення об'єктів

На 71 рядку використовується глобальна для класу менеджера змінна $creationStack, яка є буфером для уникнення циклічної залежності.

Циклічна залежність (англ. Cyclomatic dependency) – логічна помилка, яка виникає тоді, коли серед аргументів створюваного об'єкта класу є клас, в аргументах конструктора якого перебуває залежність на створюваний клас (рис. 9).

Обробка спроєктованих таким чином класів будь-яким кодом автоматичного створення об'єктів залежностей приведе до нескінченної рекурсії та врешті-решт закінчиться «зависанням» сервера через вичерпання його ресурсів.

У циклі (із 74 рядка по 80) перевіряється можливість циклічної залежності та, якщо її не виявлено, викликається метод отримання (рис. 4) об'єкта для кожного класу з аргументу конструктора.

```php
class A
{
    private B $b;

    public function __construct(B $b)
    {
        $this->b = $b;
    }
}

class B
{
    private A $a;

    public function __construct(A $a)
    {
        $this->a = $a;
    }
}
```

Рис. 9. Приклад циклічної залежності

Таким чином отримується рекурсія створення нових об'єктів, які необхідні для класу, та унеможливлюється створення копій об'єктів, які вже наявні в системі. На останніх рядках виключається перевірка циклічної залежності та створюється об'єкт класу з передачею всіх необхідних аргументів. Тепер для перевірки роботи менеджера об'єктів слід створити такий набір класів:

– клас HelloWorld (рис. 12), який виводитиме текст «Hello World»;
– клас Writer (рис. 11), який виконує роль записувача тексту;
– клас Dictionary (рис. 10), роль якого – надання необхідного тексту.

Для створення об'єкта кожного класу необхідно передати до нього попередньо створений об'єкт залежного класу.

```php
<?php
declare(strict_types=1);

namespace Framework\Test;

class Dictionary
{
    const TEXT = "Hello World";

    /**
     * @return string
     */
    public function getText(): string
    {
        return self::TEXT;
    }
}
```

Рис. 10. Клас словника

```php
<?php
declare(strict_types=1);

namespace Framework\Test;

class Writer
{
    /**
     * @var Dictionary
     */
    private Dictionary $dictionary;

    /**
     * @param Dictionary $dictionary
     */
    public function __construct(Dictionary $dictionary)
    {
        $this->dictionary = $dictionary;
    }

    /**
     * Writes text from dictionary
     */
    public function write(): void
    {
        echo $this->dictionary->getText();
    }
}
```

Рис. 11. Клас записувача тексту

```php
<?php
declare(strict_types=1);

namespace Framework\Test;

class HelloWorld
{
    /**
     * @var Writer
     */
    private Writer $writer;

    /**
     * @param Writer $writer
     */
    public function __construct(Writer $writer)
    {
        $this->writer = $writer;
    }

    /**
     * Use writer to write text
     */
    public function helloWorld(): void
    {
        $this->writer->write();
    }
}
```

Рис. 12. Основний клас перевірки

Без використання менеджера об’єктів за тим же проєктуванням довелося б створювати все вручну. Тепер слід додати ще декілька рядків в основний файл сервера для перевірки дії менеджера об’єктів (рис. 13).

```php
<?php
define('DS', DIRECTORY_SEPARATOR);
define('SOURCE_PATH', 'src');
require 'autoload.php';

use Framework\ObjectManager;

$objectManager = ObjectManager::getInstance();

/** @var \Framework\Test\HelloWorld $helloWorld */
$helloWorld = $objectManager->get(\Framework\Test\HelloWorld::class);

$helloWorld->helloWorld();
```

Рис. 13. Код перевірки роботи менеджера об’єктів

Насамкінець слід викликати метод зі створеного об’єкта й отримати результат (рис. 14).

```
a1@1s-MacBook-Pro-4 untitled % php7.4 index.php
Hello World%
a1@1s-MacBook-Pro-4 untitled %
```

Рис. 14. Результат тестування

Наприкінці необхідно зауважити, що як і в будь-якому шаблоні програмування у впровадженні залежностей є свої переваги та недоліки (Контейнер внедрения зависимостей (DI) в PHP). Щодо переваг, то можемо зазначити, зокрема, такі:

– швидке створення комплекту об’єктів для утворення будь-якого об’єкта в системі;
– зручне проєктування класів системи;
– використання раніше створених об’єктів, виключаючи їх дублювання;
– можливість комбінувати з майже будь-яким шаблоном.
– Щодо недоліків, то можемо навести, зокрема, такі:
– значні вимоги до правильності проєктування нових класів системи;
– за умови неуважного використання дуже просто зробити логічну помилку;
– швидкодія знижується, якщо класів, що оброблюються, стає занадто багато (наприклад, більше ніж 5000 одиниць);
– можливість у разі необхідності спеціально створювати копії об’єктів.

Висновки. Проаналізовано наявні проблеми та перспективи застосування підходу до впровадження залежностей під час розробки складних систем, у тому числі й вебзастосунків.

Ураховуючи результати проведеного аналізу, розроблено систему автоматизації впровадження залежностей, яка має важливе значення для розв'язання проблем підвищення ефективності процесів розробки вебзастосунків мовою PHP.

Таким чином шаблон програмування «Впровадження залежності» (як описане вище рішення) можна інтегрувати в будь-який новий проєкт без значної зміни його логіки. З цим шаблоном зручно працювати під час розробки рішень, що на ньому базуються.

СПИСОК ПОСИЛАНЬ

Инверсия и внедрение зависимостей, 2021. *Distillery Tech,* [online] 10 февраля 2021. Доступно: <https://webdevblog.ru/inversiya-i-vnedrenie-zavisimostej/> [Дата обращения 29 сентября 2021].

Контейнер внедрения зависимостей (DI) в PHP. *Русские Блоги*. [online] Доступно: <> [Дата обращения 03 октября 2021].

PHP Инъекция Зависимостей. *CodeRoad*, [online] 08 апреля 2012. Доступно: <https://russianblogs.com/article/29418350> [Дата обращения 01 октября 2021].

Русаков, М., 2017. Что такое внедрение зависимостей в PHP. *MyRusakov.ru,* [online] 15 августа 2017. Доступно: <https://myrusakov.ru/php-dependency-injection.html> [Дата обращения 01 октября 2021].

Что такое внедрение зависимостей и как это использовать в разработке? *AppTractor,* [online] 12 января 2021. Доступно: <https://apptractor.ru/info/articles/dependency-injection.html> [Дата обращения 02 октября 2021].

Dependency Injection (Внедрение зависимостей). *PHP Portal*. [online] Available at: <https://www.kobzarev.com/programming/di/> [Дата обращения 03 октября 2021].

Kukurba, V., 2019. Dependency Injection and Inversion of Control in JavaScript. *Medium*, [online] 8 September 2019. Available at: <https://viktor-kukurba.medium.com> [Accessed 28 September 2021].

Kulkarni, K. ES3 ECMAScript (ES7 + ES8 + ES9 + ES10) New Features – Javascrip. *Blog Post*. [online] Available at: <https://www.cronj.com/blog/javascript-es7-es8-new-features/> [Accessed 02 October 2021].

REFERENCES

Chto takoe vnedrenie zavisimostei i kak eto ispolzovat v razrabotke? [What is Dependency Injection and how can I use it in development?]. *AppTractor*, [online] 12 January 2021. Available at: <https://apptractor.ru/info/articles/dependency-injection.html> [Accessed 02 October 2021].

Dependency Injection (Vnedrenie zavisimostei). *PHP Portal*. [online] Available at: <https://www.kobzarev.com/programming/di/> [Accessed 03 October 2021].

Inversiia i vnedrenie zavisimostei [Dependency Inversion and Injection], 2021. *Distillery Tech*, [online] 10 February 2021. Available at: <https://webdevblog.ru/inversiya-i-vnedrenie-zavisimostej/> [Accessed 29 September 2021].

Konteiner vnedreniia zavisimostei (DI) v PHP [PHP Dependency Injection (DI) container]. *Russian Blogs*. [online] Available at: <https:// russianblogs.com/article/294183505/> [Accessed 03 October 2021].
Kukurba, V., 2019. Dependency Injection and Inversion of Control in JavaScript. *Medium*, [online] 8 September 2019. Available at: <https://viktor-kukurba.medium.com> [Accessed 28 September 2021].
Kulkarni, K. ES3 ECMAScript (ES7 + ES8 + ES9 + ES10) New Features – Javascrip. *Blog Post*. [online] Available at: <https://www.cronj.com/blog/javascript-es7-es8-new-features/> [Accessed 02 October 2021].
PHP Inektciia Zavisimostei [PHP Dependency Injection]. *CodeRoad*, [online] 08 April 2012. Available at: <https://coderoad.ru/1006 4970/ PHP-Инъекция-Зависимостей> [Accessed 01 October 2021].
Rusakov, M., 2017. Chto takoe vnedrenie zavisimostei v PHP [What is Dependency Injection in PHP]. *MyRusakov.ru*, [online] 15 August 2017. Available: <https://myrusakov.ru/php-dependency-injection.html> [Accessed 01 October 2021].

УДК 004.65:004.9

Tkachenko Olha,
PhD in Physical and Mathematical Sciences,
Associate Professor, Department of Information Technologies and Design,
State University of Infrastructure and Technology,
Kyiv, Ukraine
oitkachen@gmail.com
https://orcid.org/0000-0003-1800-618X

Rusakov Mykyta,
Master's student,
Department of Information Technologies and Design,
State University of Infrastructure and Technology,
Kyiv, Ukraine
zeeynnik@gmail.com
https://orcid.org/0000-0002-0248-2520

SOME ASPECTS OF AUTOMATED IMPLEMENTATION OF DEPENDENCIES IN PHP LANGUAGE

The purpose of the article is to research, analyze and consider the general problems and prospects of using the principle of implementing dependencies in the development of web applications in the PHP language.

Research methods are methods of semantic analysis of the basic concepts of this subject area (web technologies and web applications). The article considers the existing approaches to control inversion based on the use of the dependency implementation template.

The novelty of the research is to solve the control inversion problems based on a template when designing web applications.

The conclusion of the research conducted in the article is that the existing problems and prospects for the application of the principle of dependency implementation in web

applications with program code written in PHP have been analyzed. Taking into account the results of the analysis, the authors have developed a system for automating the implementation of dependencies, which is important for solving problems of improving the efficiency of web application development processes in PHP.

Keywords: web application; web application design; dependency implementation; PHP; ECMAScript; JavaScript.

УДК 004.65:004.9

Ткаченко Ольга,
кандидат физико-математических наук,
доцент кафедры информационных технологий и дизайна,
Государственный университет инфраструктуры и технологий,
Киев, Украина
oitkachen@gmail.com
https://orcid.org/0000-0003-1800-618X

Русаков Никита,
магистрант, кафедра информационных технологий и дизайна,
Государственный университет инфраструктуры и технологий,
Киев, Украина
zeeynnik@gmail.com
https://orcid.org/0000-0002-0248-2520

НЕКОТОРЫЕ АСПЕКТЫ АВТОМАТИЗИРОВАННОГО ВНЕДРЕНИЯ ЗАВИСИМОСТЕЙ В ЯЗЫКЕ PHP

Целью статьи является исследование, анализ и рассмотрение общих проблем и перспектив использования принципа внедрения зависимостей при разработке веб-приложений на языке PHP.

Методами исследования являются методы семантического анализа основных понятий данной предметной области (веб-технологии и веб-приложения). В статье рассмотрены существующие подходы к инверсии управления на основе использования шаблона внедрения зависимостей.

Новизной проведенного исследования является решение проблем инверсии управления на основе шаблона при проектировании веб-приложений.

Выводы. Проанализированы существующие проблемы и перспективы применения принципа внедрения зависимостей в веб-приложениях, имеющих программный код, написанный на языке PHP. Учитывая результаты проведенного анализа, разработана система автоматизации внедрения зависимостей, которая имеет важное значение для решения проблем повышения эффективности процессов разработки веб-приложений на языке PHP.

Ключевые слова: веб-приложение; проектирование веб-приложений; внедрение зависимостей; PHP; ECMAScript; JavaScript.

16.10.2021