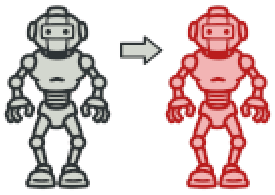

Prototype en PHP

Prototype es un patrón de diseño creacional que permite la clonación de objetos, incluso los complejos, sin acoplarse a sus clases específicas.

Todas las clases prototipo deben tener una interfaz común que haga posible copiar objetos incluso si sus clases concretas son desconocidas. Los objetos prototipo pueden producir copias completas, ya que los objetos de la misma clase pueden acceder a los campos privados de los demás.

Aprende más sobre el patrón Prototype →

Navegación

Complejidad: ★☆☆

Popularidad: ★★☆

Ejemplos de uso: El patrón Prototype está disponible en PHP **listo para usarse**. Puedes utilizar la palabra clave `clone` para crear un copia exacta de un objeto. Para añadir soporte de clonación a una clase, debes implementar un método `__clone`.

Ejemplo conceptual

Este ejemplo ilustra la estructura del patrón de diseño **Prototype** y se centra en las siguientes preguntas:

- ¿De qué clases se compone?
- ¿Qué papeles juegan esas clases?
- ¿De qué forma se relacionan los elementos del patrón?

Después de conocer la estructura del patrón, será más fácil comprender el siguiente ejemplo basado en un caso de uso real de PHP.

index.php: Ejemplo conceptual

```php
<?php

namespace RefactoringGuru\Prototype\Conceptual;

/**
 * The example class that has cloning ability. We'll see how the values of field
 * with different types will be cloned.
 */
class Prototype
{
    public $primitive;
    public $component;
    public $circularReference;

    /**
     * PHP has built-in cloning support. You can `clone` an object without
     * defining any special methods as long as it has fields of primitive types.
     * Fields containing objects retain their references in a cloned object.
     * Therefore, in some cases, you might want to clone those referenced
     * objects as well. You can do this in a special `__clone()` method.
     */
    public function __clone()
    {
        $this->component = clone $this->component;

        // Cloning an object that has a nested object with backreference
        // requires special treatment. After the cloning is completed, the
        // nested object should point to the cloned object, instead of the
```

```php
            $this->circularReference->prototype = $this;
        }
    }

    class ComponentWithBackReference
    {
        public $prototype;

        /**
         * Note that the constructor won't be executed during cloning. If you have
         * complex logic inside the constructor, you may need to execute it in the
         * `__clone` method as well.
         */
        public function __construct(Prototype $prototype)
        {
            $this->prototype = $prototype;
        }
    }

    /**
     * The client code.
     */
    function clientCode()
    {
        $p1 = new Prototype();
        $p1->primitive = 245;
        $p1->component = new \DateTime();
        $p1->circularReference = new ComponentWithBackReference($p1);

        $p2 = clone $p1;
        if ($p1->primitive === $p2->primitive) {
            echo "Primitive field values have been carried over to a clone. Yay!\n";
        } else {
            echo "Primitive field values have not been copied. Booo!\n";
        }
        if ($p1->component === $p2->component) {
            echo "Simple component has not been cloned. Booo!\n";
        } else {
            echo "Simple component has been cloned. Yay!\n";
        }

        if ($p1->circularReference === $p2->circularReference) {
            echo "Component with back reference has not been cloned. Booo!\n";
        } else {
            echo "Component with back reference has been cloned. Yay!\n";
        }

        if ($p1->circularReference->prototype === $p2->circularReference->prototype) {
            echo "Component with back reference is linked to original object. Booo!\n";
        } else {
}

clientCode();
```

Output.txt: Resultado de la ejecución

```
Primitive field values have been carried over to a clone. Yay!
Simple component has been cloned. Yay!
Component with back reference has been cloned. Yay!
Component with back reference is linked to the clone. Yay!
```

Ejemplo del mundo real

El patrón **Prototype** proporciona una forma cómoda de replicar objetos existentes en lugar de intentar reconstruirlos copiando directamente todos sus campos. La vía directa no solo te acopla a las clases de los objetos clonados, sino que además no te permite copiar los contenidos de los campos privados. El patrón Prototype te permite realizar la clonación dentro del contexto de la clase clonada, donde el acceso a los campos privados de la clase no está restringido.

Este ejemplo te muestra cómo clonar un objeto Page (Página) complejo utilizando el patrón Prototype. La clase Page tiene muchos campos privados, que se trasladarán al objeto clonado gracias al patrón Prototype.

index.php: Ejemplo del mundo real

```php
<?php

namespace RefactoringGuru\Prototype\RealWorld;

/**
 * Prototype.
 */
class Page
{
    private $title;

    private $body;

    /**
    private $author;

    private $comments = [];

    /**
     * @var \DateTime
     */
    private $date;

    // +100 private fields.

    public function __construct(string $title, string $body, Author $author)
    {
        $this->title = $title;
        $this->body = $body;
        $this->author = $author;
        $this->author->addToPage($this);
        $this->date = new \DateTime();
    }

    public function addComment(string $comment): void
    {
        $this->comments[] = $comment;
    }

    /**
     * You can control what data you want to carry over to the cloned object.
     *
     * For instance, when a page is cloned:
     * - It gets a new "Copy of ..." title.
     * - The author of the page remains the same. Therefore we leave the
     * reference to the existing object while adding the cloned page to the list
     * of the author's pages.
     * - We don't carry over the comments from the old page.
     * - We also attach a new date object to the page.
     */
    public function __clone()
    {
        $this->title = "Copy of " . $this->title;
        $this->author->addToPage($this);
        $this->comments = [];
        $this->date = new \DateTime();
    }
}

class Author
{
    private $name;

    /**
    private $pages = [];

    public function __construct(string $name)
    {
        $this->name = $name;
    }

    public function addToPage(Page $page): void
    {
        $this->pages[] = $page;
    }
}

/**
 * The client code.
 */
function clientCode()
{
    $author = new Author("John Smith");
    $page = new Page("Tip of the day", "Keep calm and carry on.", $author);

    // ...

    $page->addComment("Nice tip, thanks!");

    // ...

    $draft = clone $page;
    echo "Dump of the clone. Note that the author is now referencing two objects.\n\n";
    print_r($draft);
}

clientCode();
```

Output.txt: Resultado de la ejecución

```
Dump of the clone. Note that the author is now referencing two objects.

RefactoringGuru\Prototype\RealWorld\Page Object
(
    [title:RefactoringGuru\Prototype\RealWorld\Page:private] => Copy of Tip of the day
    [body:RefactoringGuru\Prototype\RealWorld\Page:private] => Keep calm and carry on.
    [author:RefactoringGuru\Prototype\RealWorld\Page:private] => RefactoringGuru\Prototyp
        (
            [name:RefactoringGuru\Prototype\RealWorld\Author:private] => John Smith
            [pages:RefactoringGuru\Prototype\RealWorld\Author:private] => Array
```

```
                                (
                                    [title:RefactoringGuru\Prototype\RealWorld\Page:private] => T
                                    [body:RefactoringGuru\Prototype\RealWorld\Page:private] => Ke
                                    [author:RefactoringGuru\Prototype\RealWorld\Page:private] => 
 *RECURSION*
                                    [comments:RefactoringGuru\Prototype\RealWorld\Page:private] =
                                        (
                                            [0] => Nice tip, thanks!
                                        )

                                    [date:RefactoringGuru\Prototype\RealWorld\Page:private] => Da
                                        (
                                            [date] => 2018-06-04 14:50:39.306237
                                            [timezone_type] => 3
                                            [timezone] => UTC
                                        )

                                )

                            [1] => RefactoringGuru\Prototype\RealWorld\Page Object
 *RECURSION*
                        )

                )

            [comments:RefactoringGuru\Prototype\RealWorld\Page:private] => Array
                (
                )

            [date:RefactoringGuru\Prototype\RealWorld\Page:private] => DateTime Object
                (
                    [date] => 2018-06-04 14:50:39.306272
                    [timezone_type] => 3
                    [timezone] => UTC
                )

        )
```

Ejemplo conceptual | Ejemplo del mundo real

LEER SIGUIENTE

Ukrainian office:
FOP Olga Skobeleva
Abolmasova 7
Kyiv, Ukraine, 02002
Email: support@refactoring.guru

Spanish office:
Oleksandr Shvets
Avda Pamplona 63, 4b
Pamplona, Spain, 31010
Email: spain@refactoring.guru