PHP-дегі Полиморфизм

Орындаған: Бирюков Максим

Тексерген: Сейілхан Айтжан

Топ: ETB-0923-2

Кіріспе

Полиморфизм - объектілі-бағдарланған бағдарламалаудағы негізгі принциптердің бірі. Ол әртүрлі объектілерге бірдей әдісті қолдануға мүмкіндік береді, бірақ әр объект әдісті өзіне тән жолмен жүзеге асырады. PHP-де полиморфизм ***Мұрагерлік***, ***интерфейстер*** және ***абстракт*** кластар арқылы қолданылады.

Мысалы: егер біз әртүрлі жануарлардың дыбысын шығаратын бағдарламаны жазғымыз келсе, әр жануар үшін жеке код жазудың орнына, бір әдісті әртүрлі класстарда қайта қолдана аламыз.

Полиморфизм дегеніміз не

Анықтама: бірдей әдісті әртүрлі объектілер әртүрлі жүзеге асыра алады.

Түрлері:

Подтиптік полиморфизм - бір ата-аналық кластан туындаған объектілер.

Параметрлік полиморфизм - әдістер әртүрлі типтегі деректермен жұмыс істей алады.

Перегрузка әдістері - PHP-де нақты қолдауы жоқ, бірақ шартты түрде жүзеге асыруға болады.

Мұрагерлік арқылы полиморфизм

```php
<?php
class Animal {
    public function makeSound() {
        echo "Дауыс\n";
    }
}

class Dog extends Animal {
    public function makeSound() {
        echo "Шар\n";
    }
}

class Cat extends Animal {
    public function makeSound() {
        echo "Мысық\n";
    }
}

function playSound(Animal $animal) {
    $animal->makeSound();
}

playSound(new Dog()); // Шар
playSound(new Cat()); // Мысық
?>
```

Түсініктеме:

playSound функциясы әртүрлі жануар объектілерімен жұмыс істей алады. Бұл - подтиптік полиморфизм мысалы.

Интерфейстер арқылы полиморфизм

```php
<?php
interface Logger {
    public function log(string $message);
}

class FileLogger implements Logger {
    public function log(string $message) {
        echo "Файлға жазылды: $message\n";
    }
}

class DatabaseLogger implements Logger {
    public function log(string $message) {
        echo "Дерекқорға жазылды: $message\n";
    }
}

function writeLog(Logger $logger, string $message) {
    $logger->log($message);
}

writeLog(new FileLogger(), "Сынақ хабар");
writeLog(new DatabaseLogger(), "Сынақ хабар");
?>
```

Түсініктеме:

Интерфейсті жүзеге асыратын кез келген класс writeLog функциясында қолданылуы мүмкін. Бұл әдіс кодтың икемділігін арттырады.

Абстракт кластар арқылы полиморфизм

```php
<?php
abstract class Shape {
    abstract public function area();
}


class Circle extends Shape {
    private $radius;
    public function __construct($radius) {
        $this->radius = $radius;
    }
    public function area() {
        return pi() * $this->radius ** 2;
    }
}
```

```php
class Rectangle extends Shape {
    private $width, $height;
    public function __construct($width, $height) {
        $this->width = $width;
        $this->height = $height;
    }
    public function area() {
        return $this->width * $this->height;
    }
}


$shapes = [new Circle(5), new Rectangle(4, 6)];


foreach ($shapes as $shape) {
    echo $shape->area() . "\n";
}
?>
```

Түсініктеме:

Әртүрлі фигуралар area әдісін әртүрлі есептейді, бірақ олармен жұмыс бірдей. Бұл абстракт кластар арқылы полиморфизм мысалы.

Полиморфизмнің артықшылықтары

Кодтың икемділігі: бір функция бірнеше объектімен жұмыс істей алады.

Қайта қолданылуы: кодты бірнеше жерде қайта жазбай қолдануға болады.

Қолдау жеңілдігі: жаңа класс қосқанда ескі кодты өзгерту қажет емес.

Жаңа функционал қосу: жүйеге жаңа функцияларды оңай енгізуге болады.

Практикалық қолдану салалары

Логтау (Logging): әртүрлі логгерлерді бір интерфейс арқылы қолдану.

Объектілермен жұмыс: әртүрлі объектілерді бір функция арқылы өңдеу.

Жобалау үлгілері: Strategy, Factory, Observer.

Қысқа қосымша мысалдар

```php
<?php
function printData(array|string $data) {
    if (is_array($data)) {
        foreach ($data as $item) echo $item . "\n";
    } else {
        echo $data . "\n";
    }
}

printData("Сәлем");
printData(["Бір", "Екі", "Үш"]);
?>
```

Түсініктеме:

Бір әдіс әртүрлі типтегі деректерді басып шығара алады.

Перегрузка имитациясы

```php
<?php
class Calculator {
    public function sum(...$numbers) {
        return array_sum($numbers);
    }
}

$calc = new Calculator();
echo $calc->sum(1, 2);       // 3
echo $calc->sum(1, 2, 3, 4); // 10
?>
```

Түсініктеме:

PHP-де нақты перегрузка жоқ, бірақ ...$numbers арқылы әртүрлі аргументтерді қабылдауға болады.

Полиморфизм қайда қолданылады

Полиморфизмді тек бағдарламалаудағы теория деп ойлауға болмайды. Ол нақты өмірде де бар.

Мысалы:

Бізде ***“көлік”*** деген ұғым бар. Көліктің түрлері көп - жеңіл көлік, автобус, мотоцикл. Барлығы қозғала алады, бірақ әрқайсысы өз әдісімен жүреді.

Бағдарламада да осылай - бірдей әдіс (мысалы, **move()** немесе start()) әртүрлі объектілерде әртүрлі жұмыс істейді.

Полиморфизм қайда қолданылады

```
<?php
class Vehicle {
    public function move() {
        echo "Көлік қозғалды.\n";
    }
}

class Car extends Vehicle {
    public function move() {
        echo "Машина жолмен
жүріп барады.\n";
    }
}

class Bike extends Vehicle {
    public function move() {
        echo "Мотоцикл қозғалып келеді.\
n";
    }
}

$vehicles = [new Car(), new Bike()];
foreach ($vehicles as $v) {
    $v->move();
}
?>
```

Түсініктеме:

Бірдей move() әдісі бар, бірақ әр көлік оны өз тәсілімен орындайды.

Неліктен полиморфизм маңызды

Полиморфизм бағдарламаны жеңіл әрі ыңғайлы етеді.

Төмендегі себептер соны дәлелдейді:

Код қысқарады. Бір функция арқылы бірнеше объектіні өңдей аламыз.

Түсінікті болады. Бағдарламаны оқыған адамға логика бірден көрінеді.

Қайта жазудың қажеті жоқ. Егер жаңа класс қосылса, ескі код өзгермейді.

Жүйе кеңейеді. Керек болса, жаңа объект қосып, бәрін өзгеріссіз қалдыруға болады.

Неліктен полиморфизм маңызды

Қарапайым мысал:

Мектеп жүйесінде мұғалімдер мен оқушылар бар. Екеуі де “адам” деген ортақ класстан туындайды.

```php
<?php
class Human {
    public function info() {
        echo "Бұл адам.\n";
    }
}

class Teacher extends Human {
    public function info() {
        echo "Бұл мұғалім.\n";
    }
}

class Student extends Human {
    public function info() {
        echo "Бұл оқушы.\n";
    }
}

$people = [new Teacher(), new Student()];
foreach ($people as $p) {
    $p->info();
}
?>
```

Нәтиже:

Бағдарлама “адам” типін ғана біледі, бірақ нақты кім екенін (мұғалім не оқушы) өзі анықтайды.

Қорытынды

Полиморфизм - объектілі-бағдарланған бағдарламалаудағы маңызды принцип. Ол бірдей әдісті әртүрлі объектілерде әртүрлі орындауға мүмкіндік береді. PHP-де ол мұрагерлік, интерфейстер және абстракт кластар арқылы іске асады. Бұл тәсіл кодты қысқартып, түсінікті және кеңейтуге ыңғайлы етеді. Полиморфизм бағдарламаны икемді әрі тиімді қылады.

Пайдаланылған әдебиеттер

1. **Википедия**. https://ru.wikipedia.org
2. **PHP.net** https://www.php.net/
3. **Tproger.ru.** https://tproger.ru