PHP

Web Programming: Advance
(Integrative-Generative AI Edition)

Kew Contents:

- Object-Oriented Programming (OOP) in PHP
- File and Filesystem Handling
- Professional Error and Exception Management
- Introduction to Web Application Development using PHP Framework

Bibliography

Student Price Book Center

คำนำ

ในปัจจุบัน การพัฒนาเว็บแอปพลิเคชันด้วยภาษา PHP ได้พัฒนาไปไกลกว่าการเขียนโค้ดแบบพื้นฐาน โดยมีความจำเป็นต้องอาศัยแนวคิดและเทคนิคที่ซับซ้อนขึ้น เพื่อรองรับระบบที่มีผู้ใช้งานจำนวนมาก มีโครงสร้างซับซ้อน และต้องการความปลอดภัยในระดับองค์กร หนังสือ *PHP Web Programming: Advance* เล่มนี้ถูกจัดทำขึ้นเพื่อเป็นคู่มือการเรียนรู้ในระดับสูงที่ต่อยอดจากพื้นฐานเดิม ครอบคลุมหัวข้อสำคัญด้านการเขียนโปรแกรมเชิงวัตถุ การจัดการไฟล์และระบบไฟล์ การรับมือกับข้อผิดพลาดและข้อยกเว้น รวมถึงการพัฒนาเว็บแอปพลิเคชันด้วย PHP Framework ซึ่งทั้งหมดล้วนเป็นองค์ความรู้ที่จำเป็นสำหรับนักพัฒนายุคใหม่

เริ่มต้นที่ **บทที่ 9: การเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP** ซึ่งเป็นพื้นฐานสำคัญของการเขียนโปรแกรมสมัยใหม่ ผู้อ่านจะได้เข้าใจแนวคิดของ Class, Object, Property และ Method ตลอดจนหลักการ Encapsulation, Inheritance และ Polymorphism ที่ช่วยให้โค้ดมีโครงสร้างที่ดี ยืดหยุ่น และนำกลับมาใช้ใหม่ได้ นอกจากนี้ยังครอบคลุมการใช้งาน Interface, Abstract Class, Traits และ Namespaces ซึ่งเป็นฟีเจอร์เฉพาะของ PHP ที่ช่วยให้ระบบมีความเป็นโมดูลและลดความซับซ้อนของโค้ดในโปรเจกต์ขนาดใหญ่

บทที่ 10: การจัดการไฟล์และระบบไฟล์ เน้นการประยุกต์ใช้คำสั่งพื้นฐานของ PHP ในการอ่านและเขียนไฟล์ ไม่ว่าจะเป็นแบบ Text หรือ Binary พร้อมอธิบายการจัดการไฟล์ที่อัปโหลดจากฟอร์ม HTML อย่างปลอดภัย การตรวจสอบประเภทไฟล์ และการออกแบบระบบจัดเก็บไฟล์ที่มีประสิทธิภาพ ผู้อ่านจะได้เรียนรู้เทคนิคในการทำงานกับระบบไฟล์อย่างปลอดภัย รองรับการใช้งานจริง เช่น ระบบจัดเก็บเอกสาร ระบบแนบไฟล์ หรือระบบส่งข้อมูลออนไลน์

ใน **บทที่ 11: การจัดการ Error และ Exception อย่างมืออาชีพ** หนังสือได้นำเสนอแนวทางการจัดการข้อผิดพลาดด้วยกลไก try-catch-finally พร้อมเทคนิคการสร้าง Custom Exception เพื่อแยกประเภทข้อผิดพลาดให้เหมาะสมกับบริบทที่เกิดขึ้น รวมถึงการตั้งค่า Error Reporting การ Logging และการออกแบบระบบ Log ที่สามารถใช้ตรวจสอบปัญหาในระบบจริงได้อย่างมีประสิทธิภาพ ทั้งหมดนี้มีบทบาทสำคัญในการทำให้เว็บแอปพลิเคชันมีความเสถียร ปลอดภัย และพร้อมใช้งานในสภาพแวดล้อม production

ในส่วนของ **บทที่ 12: การพัฒนาเว็บแอปพลิเคชันด้วย PHP Framework เบื้องต้น** ผู้อ่านจะได้ทำความรู้จักกับ Framework ยอดนิยม เช่น Laravel, Symfony และ CodeIgniter ซึ่งช่วยจัดการโครงสร้างระบบให้เป็นระเบียบ และเร่งกระบวนการพัฒนาอย่างเป็นระบบ บทนี้ครอบคลุมตั้งแต่การติดตั้งโปรเจกต์ การกำหนด Routing การใช้งาน Controllers และ Views การทำงานร่วมกับ ORM (Eloquent/Doctrine) ตลอดจนการจัดการ Migration และ Seed Database โดยเนื้อหาเน้นการลงมือทำจริง เพื่อให้ผู้อ่านสามารถพัฒนาแอปพลิเคชันได้อย่างมีประสิทธิภาพ

หนังสือเล่มนี้ได้รับการออกแบบโดยคำนึงถึงลำดับการเรียนรู้ของผู้อ่าน โดยแต่ละบทมีการไล่ระดับความซับซ้อนจากพื้นฐานสู่การประยุกต์ใช้ พร้อมด้วยตัวอย่างโค้ดที่สามารถทดลองใช้งานได้จริง

เพื่อให้ผู้อ่านเข้าใจแนวคิดอย่างเป็นระบบ และสามารถนำไปประยุกต์ใช้ในงานจริงได้อย่างมั่นใจ โดยเฉพาะในโปรเจกต์ที่ต้องการโครงสร้างชัดเจน และมาตรฐานระดับมืออาชีพ

จุดเด่นของหนังสือเล่มนี้คือการรวมองค์ความรู้ที่จำเป็นต่อการพัฒนาเว็บด้วย PHP ในระดับสูง โดยไม่จำกัดเพียงแนวคิดหรือทฤษฎี แต่ให้ความสำคัญกับการใช้งานจริง และแนวทางที่สามารถนำไปใช้ต่อยอดในองค์กรได้ ไม่ว่าจะเป็นการสร้างระบบหลังบ้าน การจัดการผู้ใช้ ระบบอัปโหลดไฟล์ ระบบ log หรือแม้แต่เว็บแอปพลิเคชันเชิงธุรกิจเต็มรูปแบบ

สุดท้ายนี้ ผู้เขียนหวังเป็นอย่างยิ่งว่า *PHP Web Programming: Advance* จะเป็นแนวทางสำคัญสำหรับผู้ที่ต้องการก้าวสู่การเป็นนักพัฒนา PHP มืออาชีพ ไม่เพียงในด้านทักษะเทคนิคเท่านั้น แต่รวมถึงการออกแบบระบบให้มีประสิทธิภาพ มีความปลอดภัย และสามารถรองรับการเปลี่ยนแปลงในอนาคตได้อย่างยั่งยืน

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 9
การเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP
(OOP by PHP)

เนื้อหา

- การเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP
- รายละเอียดเชิงลึกของการเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP
- แนวคิด OOP (Class, Object, Property, Method)
- การสืบทอด (Inheritance) และ Polymorphism
- Encapsulation และ Access Modifiers (public, protected, private)
- Interface และ Abstract Class
- การทำงานกับ Traits และ Namespaces ใน PHP
- ตัวอย่างโปรแกรม PHP แบบบูรณาการ

บทที่ 9: การเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP

การพัฒนาโปรแกรมเชิงวัตถุ (Object-Oriented Programming: OOP) ถือเป็นแนวคิดสำคัญในการออกแบบซอฟต์แวร์ที่มีความยืดหยุ่นและสามารถดูแลรักษาได้ในระยะยาว ในโลกของ PHP ซึ่งเริ่มจากภาษาเชิงโพรซีเดอร์ ได้พัฒนาไปจนสามารถรองรับหลักการ OOP อย่างเต็มรูปแบบ ตั้งแต่เวอร์ชัน 5 ขึ้นไป หนังสือเล่มนี้จึงได้นำเสนอ “บทที่ 9: การเขียนโปรแกรมเชิงวัตถุใน PHP” เพื่อให้ผู้อ่านเข้าใจแนวคิดเบื้องต้น ตลอดจนสามารถนำไปประยุกต์ใช้ได้จริงในการพัฒนาเว็บแอปพลิเคชันที่มีโครงสร้างซับซ้อน

บทเริ่มต้นด้วยการอธิบายแนวคิดพื้นฐานของ OOP ได้แก่ **คลาส (Class)** ซึ่งเปรียบเสมือนพิมพ์เขียวของวัตถุ และ **อ็อบเจ็กต์ (Object)** ที่เป็นสิ่งที่ถูกสร้างขึ้นจากคลาส รวมถึงองค์ประกอบที่สำคัญ เช่น **พร็อพเพอร์ตี (Property)** หรือคุณลักษณะของวัตถุ และ **เมธอด (Method)** หรือฟังก์ชันที่อธิบายพฤติกรรมของวัตถุนั้น เพื่อปูพื้นฐานให้ผู้อ่านสามารถเข้าใจแนวคิดได้อย่างเป็นระบบ

ต่อมาผู้อ่านจะได้เรียนรู้การใช้ **Inheritance** หรือการสืบทอด ซึ่งช่วยให้สามารถสร้างคลาสใหม่จากคลาสเดิมได้อย่างมีประสิทธิภาพ พร้อมด้วยการใช้งาน **Polymorphism** ที่ทำให้สามารถใช้เมธอดเดียวกันกับอ็อบเจ็กต์ต่างชนิดกันได้ ซึ่งเป็นหลักการที่สำคัญต่อการเขียนโปรแกรมแบบยืดหยุ่น และรองรับการเปลี่ยนแปลงในอนาคตได้ดี

Encapsulation คืออีกหนึ่งหลักการสำคัญที่ถูกกล่าวถึงในบทนี้ โดยเน้นการควบคุมการเข้าถึงข้อมูลและเมธอดภายในวัตถุผ่าน **Access Modifiers** ได้แก่ public, protected, และ private ซึ่งช่วยให้สามารถออกแบบคลาสที่ปลอดภัย และลดการพึ่งพากันระหว่างโมดูลต่าง ๆ

เพื่อให้การออกแบบระบบมีความยืดหยุ่นมากยิ่งขึ้น ผู้อ่านจะได้ทำความเข้าใจกับ **Interface** และ **Abstract Class** ซึ่งเป็นเครื่องมือที่ช่วยให้นักพัฒนาสามารถออกแบบสัญญา (contract) ของระบบได้ล่วงหน้า ส่งเสริมการทำงานแบบแยกส่วน (modular design) และการพัฒนาแบบทำงานร่วมกัน (collaborative development) ได้อย่างเป็นระบบ

นอกจากนี้ บทนี้ยังได้นำเสนอหัวข้อ **Traits** และ **Namespaces** ซึ่งเป็นฟีเจอร์เฉพาะของ PHP ที่ช่วยในการจัดการโค้ดซ้ำซ้อน และแยกพื้นที่การทำงานของคลาสหรือเมธอดให้มีความเป็นระเบียบ ลดความขัดแย้งของชื่อ (name collision) โดยเฉพาะในโปรเจกต์ขนาดใหญ่หรือโปรเจกต์ที่ต้องพึ่งพาไลบรารีจากภายนอก

บทที่ 9 จึงเป็นบทที่มีความสำคัญอย่างยิ่งสำหรับผู้เรียนที่ต้องการยกระดับจากการเขียนโค้ดแบบฟังก์ชันทั่วไปไปสู่การพัฒนาระบบเชิงโครงสร้าง ด้วยแนวคิดเชิงวัตถุที่สามารถนำไปประยุกต์ใช้ได้จริงในการสร้างเว็บแอปพลิเคชันระดับมืออาชีพ ทั้งในด้านโครงสร้าง ประสิทธิภาพ และความปลอดภัยของระบบในระยะยาว

การเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP

1. แนวคิด OOP (Class, Object, Property, Method)

- **Class** คือ แม่แบบ (template) หรือพิมพ์เขียวสำหรับสร้างวัตถุ (object) โดยกำหนดคุณสมบัติ (properties) และพฤติกรรม (methods)
- **Object** คือ อินสแตนซ์ของคลาส คือวัตถุจริงที่ถูกสร้างขึ้นมาจากคลาส
- **Property** คือ ตัวแปรภายในคลาสที่เก็บข้อมูลของ object
- **Method** คือ ฟังก์ชันภายในคลาสที่นิยามพฤติกรรมหรือการกระทำของ object

ตัวอย่างง่าย ๆ

```
class Car {
    public $color;
    public $brand;

    public function drive() {
        echo "Driving a $this->color $this->brand car.";
    }
}
```

```
$car1 = new Car();
$car1->color = "red";
$car1->brand = "Toyota";
$car1->drive(); // Driving a red Toyota car.
```

---

2. การสืบทอด (Inheritance) และการใช้งาน Polymorphism

- **Inheritance** คือการสร้างคลาสใหม่จากคลาสเดิม (parent class) เพื่อสืบทอด properties และ methods และเพิ่มหรือแก้ไขเพิ่มเติมได้
- **Polymorphism** คือความสามารถในการใช้งาน method ชื่อเดียวกัน แต่ทำงานต่างกันในคลาสต่างๆ โดยมักใช้ร่วมกับ inheritance หรือ interface

ตัวอย่าง Inheritance

```
class Vehicle {
    public function start() {
        echo "Vehicle started.";
    }
}

class Bike extends Vehicle {
    public function start() {
        echo "Bike started.";
    }
}

$bike = new Bike();
$bike->start(); // Bike started.
```

---

3. Encapsulation และ Access Modifiers (public, protected, private)

- **Encapsulation** คือการซ่อนรายละเอียดภายในของวัตถุและควบคุมการเข้าถึงข้อมูลผ่าน access modifiers
- **public** : สามารถเข้าถึงได้ทุกที่
- **protected** : สามารถเข้าถึงได้ในคลาสตัวเองและคลาสที่สืบทอด
- **private** : เข้าถึงได้แค่ภายในคลาสตัวเองเท่านั้น

ตัวอย่าง Encapsulation

```
class Person {
```

```
    private $name;

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

    public function getName() {
        return $this->name;
    }
}

$p = new Person();
$p->setName("John");
echo $p->getName(); // John
```

---

4. การใช้ Interface และ Abstract Class

- **Interface** กำหนด method ที่คลาสที่ implement ต้องมี แต่ไม่มีการกำหนด body ของ method
- **Abstract Class** เป็นคลาสที่ไม่สามารถสร้าง object ได้โดยตรง อาจมี method ที่มีหรือไม่มี implementation

ตัวอย่าง Interface

```
interface Logger {
    public function log($message);
}

class FileLogger implements Logger {
    public function log($message) {
        echo "Logging to file: $message";
    }
}
```

ตัวอย่าง Abstract Class

```
abstract class Animal {
    abstract public function makeSound();

    public function move() {
```

```
        echo "Moving...";
    }
}

class Dog extends Animal {
    public function makeSound() {
        echo "Bark";
    }
}

$dog = new Dog();
$dog->makeSound(); // Bark
$dog->move();      // Moving...
```

---

5. การทำงานกับ Traits และ Namespaces

- **Traits** คือกลไกสำหรับการ reuse โค้ดในคลาสต่างๆ โดยไม่ต้องสืบทอด (multiple inheritance ไม่ได้ใน PHP แต่ Traits ช่วยได้)
- **Namespace** คือการจัดกลุ่มคลาส, ฟังก์ชัน หรือคอนสแตนต์ เพื่อป้องกันชื่อชนกัน (name collision) ในโปรเจกต์ขนาดใหญ่

ตัวอย่าง Trait

```
trait LoggerTrait {
    public function log($msg) {
        echo "Log: $msg";
    }
}

class User {
    use LoggerTrait;
}

$user = new User();
$user->log("User created"); // Log: User created
```

ตัวอย่าง Namespace

```
namespace App\Models;
```

```
class User {
    public function info() {
        return "User info";
    }
}

// เรียกใช้งาน
$user = new \App\Models\User();
echo $user->info();
```

รายละเอียดเชิงลึกของการเขียนโปรแกรมเชิงวัตถุ (OOP) ใน PHP

---

1. แนวคิด OOP (Class, Object, Property, Method)

- **Class** คือ โครงสร้างหรือแม่แบบ (blueprint) สำหรับสร้างวัตถุ (objects) ซึ่งกำหนดว่าคุณสมบัติ (properties) และพฤติกรรม (methods) ของวัตถุนั้นเป็นอย่างไร
- **Object** คือ อินสแตนซ์ของคลาส คือหน่วยข้อมูลหรือวัตถุที่ถูกสร้างขึ้นจากคลาสหนึ่ง ๆ สามารถมีสถานะข้อมูล (state) และพฤติกรรม (behavior) ตามที่คลาสกำหนด
- **Property** คือ ตัวแปรที่เก็บข้อมูลหรือสถานะของ object ภายในคลาส
- **Method** คือ ฟังก์ชันที่เป็นพฤติกรรมหรือฟังก์ชันการทำงานของ object ภายในคลาส

หลักการสำคัญ คือ การออกแบบโปรแกรมโดยใช้วัตถุที่มีสถานะและพฤติกรรม รวมกันอยู่ในที่เดียว (Encapsulation) เพื่อให้ง่ายต่อการจัดการและบำรุงรักษา

---

2. การสืบทอด (Inheritance) และการใช้งาน Polymorphism

- **Inheritance (การสืบทอด)** ช่วยให้สามารถสร้างคลาสใหม่ (subclass หรือ child class) โดยรับคุณสมบัติและพฤติกรรมทั้งหมดจากคลาสต้นแบบ (superclass หรือ parent class) แล้วเพิ่มหรือปรับแต่งพฤติกรรมใหม่ได้
- ตัวอย่างเช่น คลาส Vehicle เป็น superclass ที่มี method start(), subclass Car และ Bike สืบทอด Vehicle และสามารถ override method start() ได้
- **Polymorphism (หลายรูปแบบ)** หมายถึงความสามารถที่ object หลายประเภทสามารถใช้ method เดียวกันได้ แต่ผลลัพธ์หรือพฤติกรรมแตกต่างกัน ขึ้นอยู่กับชนิดของ object จริงที่ถูกเรียกใช้งาน
- Polymorphism ช่วยเพิ่มความยืดหยุ่นในการเขียนโปรแกรม เช่น การส่ง object ในรูปแบบ parent class แต่เรียกใช้งาน method ของ subclass ได้

3. Encapsulation และ Access Modifiers (public, protected, private)

- **Encapsulation (การห่อหุ้ม)** คือการซ่อนรายละเอียดภายในของ object เพื่อป้องกันไม่ให้ภายนอกเข้าถึงโดยตรง แต่จะเข้าถึงข้อมูลผ่าน method ที่กำหนดไว้ เช่น getter/setter
- ช่วยควบคุมการเข้าถึงข้อมูล และรักษาความถูกต้องของข้อมูล
- **Access Modifiers** คือการกำหนดระดับการเข้าถึง property หรือ method

Modifier	การเข้าถึง
public	สามารถเข้าถึงได้จากทุกที่ทั้งภายในและภายนอกคลาส
protected	สามารถเข้าถึงได้ภายในคลาสและ subclass เท่านั้น
private	สามารถเข้าถึงได้เฉพาะภายในคลาสเท่านั้น

- การใช้ access modifiers อย่างเหมาะสมช่วยลดข้อผิดพลาดและเพิ่มความปลอดภัยของโปรแกรม

---

4. การใช้ Interface และ Abstract Class

- **Interface** คือ “สัญญา” ที่บอกว่าคลาสที่ implement interface นั้นต้องมี method ชื่ออะไรบ้าง โดย interface ไม่มีโค้ดจริง (ไม่มี method body)
- Interface ช่วยบังคับให้คลาสที่ต่างกันทำงานร่วมกันด้วย method ที่เหมือนกัน
- PHP รองรับการ implement interface หลายอันในคลาสเดียว (multiple interface inheritance)
- **Abstract Class** เป็นคลาสที่ไม่สามารถสร้าง object ได้โดยตรง อาจมี method ที่มีโค้ด (concrete method) และ method ที่ไม่มีโค้ด (abstract method)
- Abstract class ใช้สร้างโครงสร้างพื้นฐานให้ subclass ต้อง override method ที่เป็น abstract
- แตกต่างจาก interface ตรงที่ abstract class สามารถเก็บ state และมีโค้ดใน method ได้

---

5. การทำงานกับ Traits และ Namespaces

- **Traits** เป็นกลไกสำหรับ “การแชร์โค้ด” ระหว่างคลาสหลาย ๆ คลาส โดยไม่ต้องใช้ inheritance
- Traits ช่วยแก้ปัญหา “multiple inheritance” ที่ PHP ไม่รองรับโดยตรง
- Traits สามารถกำหนด method หรือ property ที่คลาสต่าง ๆ สามารถ “use” ได้
- หากมี method ซ้ำกันใน traits หลายตัว สามารถใช้ insteadof และ as เพื่อแก้ไขความขัดแย้งได้
- **Namespace** คือการจัดกลุ่มคลาส, ฟังก์ชัน หรือคอนสแตนต์ในกลุ่มชื่อเฉพาะเจาะจง
- Namespace ช่วยป้องกันปัญหาการชนกันของชื่อ (name collision) โดยเฉพาะในโปรเจกต์ขนาดใหญ่หรือเมื่อนำไลบรารี่มารวมกัน

- การใช้ namespace ทำให้สามารถใช้ชื่อคลาสเดียวกันในกลุ่ม namespace ต่างกันโดยไม่ชนกันได้
- เรียกใช้งาน namespace ได้ด้วยการใช้ use หรืออ้างอิงแบบเต็มชื่อ

แนวคิด OOP (Class, Object, Property, Method)

แนวคิด OOP (Object-Oriented Programming)

การเขียนโปรแกรมเชิงวัตถุ (OOP) เป็นแนวทางการเขียนโปรแกรมที่เน้นการใช้ **วัตถุ (Objects)** ซึ่งเป็นหน่วยข้อมูลที่รวมเอา **ข้อมูล (Properties)** และ **พฤติกรรม (Methods)** ไว้ด้วยกัน ภายใต้โครงสร้างที่เรียกว่า **คลาส (Class)**

---

1. Class (คลาส)

- คลาสเป็นแม่แบบหรือแบบแผน (blueprint) สำหรับสร้างวัตถุ (object)
- คลาสประกอบด้วย **properties** (ตัวแปรภายในคลาส) เพื่อเก็บสถานะข้อมูลของวัตถุ และ **methods** (ฟังก์ชันภายในคลาส) เพื่อกำหนดพฤติกรรมหรือการกระทำของวัตถุ
- คลาสสามารถกำหนดระดับการเข้าถึงของ properties และ methods ได้ผ่าน access modifiers เช่น public, protected, private

ตัวอย่างคลาส

```
class Person {
    public $name;     // Property
    public $age;

    public function greet() {  // Method
        echo "Hello, my name is $this->name.";
    }
}
```

---

2. Object (วัตถุ)

- วัตถุคืออินสแตนซ์ (instance) ที่ถูกสร้างจากคลาสหนึ่ง ๆ
- วัตถุมีข้อมูลสถานะเฉพาะ (properties) และสามารถเรียกใช้ฟังก์ชัน (methods) ที่กำหนดในคลาสนั้นได้
- การสร้างวัตถุใช้คำสั่ง new

ตัวอย่างสร้างวัตถุและใช้งาน

```
$person1 = new Person();
$person1->name = "Alice";
$person1->age = 25;
$person1->greet();  // Output: Hello, my name is Alice.
```

---

3. Property (คุณสมบัติ)

- Property คือ ตัวแปรที่เก็บข้อมูลหรือสถานะของวัตถุ
- Properties ถูกประกาศภายในคลาส สามารถกำหนดค่าเริ่มต้นได้ และเข้าถึงผ่านวัตถุด้วย $this->propertyName ภายในคลาส หรือ $object->propertyName ภายนอกคลาส (ถ้า access modifier เป็น public)

---

4. Method (เมธอด)

- Method คือฟังก์ชันที่ถูกประกาศภายในคลาส เพื่อกำหนดพฤติกรรมหรือการกระทำของวัตถุ
- Methods สามารถรับพารามิเตอร์และส่งค่ากลับได้
- ใน method ภายในคลาสใช้ $this เพื่ออ้างถึงวัตถุปัจจุบัน

ตัวอย่าง method รับพารามิเตอร์และ return ค่า

```
class Calculator {
    public function add($a, $b) {
        return $a + $b;
    }
}

$calc = new Calculator();
echo $calc->add(10, 20);  // Output: 30
```

สรุป

คำศัพท์	ความหมาย
Class	แบบแผนแม่แบบสำหรับสร้าง object กำหนด properties และ methods
Object	อินสแตนซ์ที่สร้างจาก class มีสถานะและพฤติกรรมตามที่ class กำหนด
Property	ตัวแปรภายใน class ที่เก็บข้อมูลสถานะของ object
Method	ฟังก์ชันภายใน class กำหนดพฤติกรรมของ object

นี่คือตัวอย่างโปรแกรม PHP แบบเต็มไฟล์ 3 โปรแกรม และแนวประยุกต์ 3 โปรแกรมที่ใช้แนวคิด OOP (Class, Object, Property, Method) พร้อมโครงสร้างและคำอธิบาย พร้อมผลการรันในแต่ละตัวอย่าง

ตัวอย่างโปรแกรมพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: คลาส Person พื้นฐาน

ไฟล์: Person.php

```
<?php
class Person {
    public $name;
    public $age;

    public function greet() {
        echo "Hello, my name is $this->name and I am $this->age years old.";
    }
}

// สร้างวัตถุ
$person1 = new Person();
$person1->name = "Alice";
$person1->age = 30;
$person1->greet();
?>
```

คำอธิบาย:

- สร้างคลาส Person มี properties name และ age
- มี method greet() แสดงข้อความทักทาย
- สร้าง object person1 ตั้งค่า properties แล้วเรียก method

ผลการรัน:

```
Hello, my name is Alice and I am 30 years old.
```

ตัวอย่างที่ 2: Calculator บวกเลข

ไฟล์: Calculator.php

```
<?php
class Calculator {
    public function add($a, $b) {
```

```
        return $a + $b;
    }
}

$calc = new Calculator();
echo "10 + 5 = " . $calc->add(10, 5);
?>
```

คำอธิบาย:

- คลาส Calculator มี method add รับพารามิเตอร์ 2 ตัวและคืนค่าผลบวก
- สร้าง object และเรียก method พร้อมแสดงผลลัพธ์

ผลการรัน:

```
10 + 5 = 15
```

ตัวอย่างที่ 3: สร้างคลาส Car พร้อม Constructor

ไฟล์: Car.php

```
<?php
class Car {
    public $brand;
    public $color;

    public function __construct($brand, $color) {
        $this->brand = $brand;
        $this->color = $color;
    }

    public function display() {
        echo "This car is a $this->color $this->brand.";
    }
}

$myCar = new Car("Toyota", "red");
$myCar->display();
?>
```

คำอธิบาย:

- คลาส Car มี constructor รับค่าแบรนด์และสีรถ
- Method display แสดงข้อมูลรถ
- สร้าง object พร้อมส่งค่า constructor แล้วแสดงผล

ผลการรัน:

This car is a red Toyota.

ตัวอย่างโปรแกรมแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 1: คลาส User กับการตรวจสอบอายุ

ไฟล์: User.php

```
<?php
class User {
    private $name;
    private $age;

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

    public function isAdult() {
        if ($this->age >= 18) {
            return true;
        } else {
            return false;
        }
    }

    public function greet() {
        if ($this->isAdult()) {
            echo "Hello, $this->name. You are an adult.";
        } else {
            echo "Hello, $this->name. You are a minor.";
        }
    }
```

```
}

$user1 = new User("Bob", 20);
$user1->greet();

echo "\n";

$user2 = new User("Charlie", 15);
$user2->greet();
?>
```

คำอธิบาย:

- ใช้ properties แบบ private พร้อม constructor
- method isAdult() ตรวจสอบอายุ
- method greet() แสดงข้อความตามอายุผู้ใช้

ผลการรัน:

Hello, Bob. You are an adult.

Hello, Charlie. You are a minor.

ตัวอย่างที่ 2: Inventory สินค้าและการคำนวณราคาสินค้าทั้งหมด

ไฟล์: Inventory.php

```
<?php
class Product {
    public $name;
    public $price;
    public $quantity;

    public function __construct($name, $price, $quantity) {
        $this->name = $name;
        $this->price = $price;
        $this->quantity = $quantity;
    }

    public function totalPrice() {
        return $this->price * $this->quantity;
```

```
    }
}

class Inventory {
    private $products = [];

    public function addProduct(Product $product) {
        $this->products[] = $product;
    }

    public function getTotalInventoryValue() {
        $total = 0;
        foreach ($this->products as $product) {
            $total += $product->totalPrice();
        }
        return $total;
    }
}

$inv = new Inventory();
$inv->addProduct(new Product("Book", 100, 3));
$inv->addProduct(new Product("Pen", 10, 10));
echo "Total Inventory Value: " . $inv->getTotalInventoryValue();
?>
```

คำอธิบาย:

- คลาส Product เก็บข้อมูลสินค้าและคำนวณราคาสินค้ารวม (totalPrice)
- คลาส Inventory เก็บรายการสินค้าเป็น array และคำนวณมูลค่ารวมทั้งหมด
- สร้างสินค้าหลายรายการและเพิ่มลงใน inventory

ผลการรัน:

```
Total Inventory Value: 400
```

ตัวอย่างที่ 3: BankAccount กับการฝาก-ถอนเงิน พร้อมตรวจสอบยอดเงิน

ไฟล์: BankAccount.php

```
<?php
```

```
class BankAccount {
    private $accountNumber;
    private $balance;

    public function __construct($accountNumber, $balance = 0) {
        $this->accountNumber = $accountNumber;
        $this->balance = $balance;
    }

    public function deposit($amount) {
        if ($amount > 0) {
            $this->balance += $amount;
            echo "Deposited $amount. New balance: $this->balance\n";
        }
    }

    public function withdraw($amount) {
        if ($amount > 0 && $amount <= $this->balance) {
            $this->balance -= $amount;
            echo "Withdrew $amount. New balance: $this->balance\n";
        } else {
            echo "Withdrawal failed. Insufficient funds or invalid amount.\n";
        }
    }

    public function getBalance() {
        return $this->balance;
    }
}

$account = new BankAccount("123456", 1000);
$account->deposit(500);
$account->withdraw(200);
$account->withdraw(2000);
```

```
echo "Final Balance: " . $account->getBalance();
?>
```

คำอธิบาย:

- คลาส BankAccount มี properties ส่วนตัวสำหรับเลขบัญชีและยอดเงิน
- มี method สำหรับฝาก (deposit) ถอน (withdraw) และตรวจสอบยอดเงิน (getBalance)
- ตรวจสอบความถูกต้องของจำนวนเงินก่อนทำธุรกรรม

ผลการรัน:

```
Deposited 500. New balance: 1500
Withdrew 200. New balance: 1300
Withdrawal failed. Insufficient funds or invalid amount.
Final Balance: 1300
```

การสืบทอด (Inheritance) และ Polymorphism

การสืบทอด (Inheritance) ใน PHP

แนวคิด

- การสืบทอดคือการสร้างคลาสใหม่ (child class หรือ subclass) จากคลาสที่มีอยู่แล้ว (parent class หรือ superclass)
- คลาสลูกจะได้รับคุณสมบัติ (properties) และพฤติกรรม (methods) ของคลาสแม่ทั้งหมด
- คลาสลูกสามารถเพิ่มหรือแก้ไข (override) method ของคลาสแม่ได้
- ช่วยให้การเขียนโค้ดเป็นระบบ มีการใช้ซ้ำ และลดความซ้ำซ้อน

การใช้งานใน PHP

```
class Animal {
    public function makeSound() {
        echo "Some generic sound";
    }
}

class Dog extends Animal {
    public function makeSound() {
        echo "Woof!";
    }
}
```

```
$animal = new Animal();
$animal->makeSound(); // Output: Some generic sound


$dog = new Dog();
$dog->makeSound();    // Output: Woof!
```

- Dog คือ subclass ที่สืบทอดจาก Animal
- Dog ทำการ override method makeSound เพื่อเปลี่ยนพฤติกรรมเฉพาะตัว

Polymorphism (พหุรูป)

แนวคิด

- Polymorphism คือความสามารถของวัตถุหลายชนิด (class หลายตัว) ที่มี interface หรือ method ชื่อเดียวกัน แต่พฤติกรรมต่างกัน
- ช่วยให้โค้ดที่ใช้วัตถุต่างชนิดกันแต่สามารถเรียกใช้งาน method เดียวกันได้อย่างยืดหยุ่น
- ใน PHP สามารถทำได้ผ่านการ override method ใน subclass หรือใช้ interface/abstract class

ตัวอย่าง Polymorphism ด้วยการ Override

```
class Animal {
    public function makeSound() {
        echo "Some generic sound";
    }
}

class Cat extends Animal {
    public function makeSound() {
        echo "Meow!";
    }
}

class Dog extends Animal {
    public function makeSound() {
        echo "Woof!";
    }
}
```

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

$cat = new Cat();
$dog = new Dog();

animalSound($cat);  // Output: Meow!
animalSound($dog);  // Output: Woof!
```

- ฟังก์ชัน animalSound รับ parameter เป็น Animal แต่สามารถใช้กับวัตถุชนิดใดก็ได้ที่สืบทอดจาก Animal
- เรียก method makeSound() ของวัตถุนั้น ๆ โดยอัตโนมัติ

Polymorphism ด้วย Interface

```
interface Shape {
    public function area();
}

class Rectangle implements Shape {
    private $width, $height;

    public function __construct($w, $h) {
        $this->width = $w;
        $this->height = $h;
    }

    public function area() {
        return $this->width * $this->height;
    }
}

class Circle implements Shape {
    private $radius;
```

```
    public function __construct($r) {
        $this->radius = $r;
    }

    public function area() {
        return pi() * pow($this->radius, 2);
    }
}

function printArea(Shape $shape) {
    echo "Area: " . $shape->area() . "\n";
}

$rect = new Rectangle(10, 5);
$circle = new Circle(7);

printArea($rect);    // Output: Area: 50
printArea($circle);  // Output: Area: 153.938...
```

- Interface Shape กำหนด method area()
- คลาส Rectangle และ Circle ต้อง implement method area()
- ฟังก์ชัน printArea ใช้ polymorphism ในการรับ parameter ที่เป็น Shape และเรียก area() ได้ทุกชนิด

สรุป

หัวข้อ	คำอธิบาย
Inheritance	สืบทอด properties และ methods จากคลาสแม่ คลาสลูกสามารถ override ได้
Polymorphism	วัตถุหลายชนิดที่มี interface หรือ method ชื่อเดียวกันแต่ทำงานต่างกัน
การใช้งานใน PHP	ใช้ extends เพื่อสืบทอด, override method และใช้ interface หรือ abstract class เพื่อ polymorphism

---

นี่คือตัวอย่างโปรแกรม PHP แบบเต็มไฟล์ จำนวน 3 โปรแกรมพื้นฐาน และ 3 โปรแกรมแนวประยุกต์ ที่ใช้หัวข้อ **การสืบทอด (Inheritance)** และ **Polymorphism** พร้อมโครงสร้าง โค้ด คำอธิบาย และผลการรัน

ตัวอย่างโปรแกรมพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: การสืบทอดและการ override method

ไฟล์: Animal.php

```
<?php
class Animal {
    public function makeSound() {
        echo "Some generic animal sound\n";
    }
}

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

$animal = new Animal();
$animal->makeSound(); // Some generic animal sound

$dog = new Dog();
$dog->makeSound();    // Woof! Woof!
?>
```

คำอธิบาย:

- คลาส Animal มี method makeSound
- คลาส Dog สืบทอดจาก Animal และ override method makeSound
- แสดงการเรียก method ของแต่ละคลาส

ผลการรัน:

```
Some generic animal sound
Woof! Woof!
```

ตัวอย่างที่ 2: Polymorphism กับฟังก์ชันรับพารามิเตอร์เป็น superclass

ไฟล์: Polymorphism.php

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

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

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

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

$cat = new Cat();
$dog = new Dog();

animalSound($cat);  // Meow
animalSound($dog);  // Woof
?>
```

คำอธิบาย:

- ฟังก์ชัน animalSound รับ parameter เป็น Animal แต่สามารถใช้กับวัตถุ subclass ใดก็ได้
- เรียก makeSound() ของวัตถุนั้น ๆ โดยใช้ polymorphism

ผลการรัน:

```
Meow
```

Woof

ตัวอย่างที่ 3: คลาสแม่และคลาสลูก พร้อมการเรียกใช้ method ของคลาสแม่ด้วย parent::

ไฟล์: ParentChild.php

```
<?php
class Vehicle {
    public function startEngine() {
        echo "Engine started\n";
    }
}

class Car extends Vehicle {
    public function startEngine() {
        echo "Car engine is warming up...\n";
        parent::startEngine();
    }
}

$car = new Car();
$car->startEngine();
?>
```

คำอธิบาย:

- คลาส Car override method startEngine และเรียก method เดิมของคลาสแม่ด้วย parent::startEngine()
- แสดงข้อความก่อนและหลังเรียก method ของคลาสแม่

ผลการรัน:

```
Car engine is warming up...
Engine started
```

ตัวอย่างโปรแกรมแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 1: ระบบสมาชิก ใช้ inheritance และ polymorphism สำหรับประเภทสมาชิกต่างกัน

ไฟล์: Member.php

```
<?php
```