PHP OOP & MVC

BEFORE WE GET STARTED

Make sure you done all of PHP MYSQL exercise at

https://github.com/namnh06/aptech-php-course

Searching on Google : PHP OOP.

OOP

- OOP stand for Object-Oriented Programming.
- A programming language model organized around objects rather than “actions” and data rather than logic.
- There are 4 major principles of Object-Oriented Programming : Inheritance, Polymorphism, Encapsulation and Abstraction.
- Easy to learn, hard to handle.

CLASS

- This is a programmer-defined data type, which includes local functions as well as local data.
- It's a template for making many instances of the same kind (or class) of object.
- Class has properties and methods.
- Magic methods can do something useful.

```php
<?php
class Hero
{
    protected $name;

    public function __construct($name = "Undefined")
    {
        echo "Begin of class<br>";
        $this->name = $name;
    }

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

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

    public function display()
    {
        echo $this->name;
    }

    public function __destruct()
    {
        echo "<br>End of class";
    }

}
```

VISIBILITY

- Each method and property has its visibility.
- Three types of visibility declared by keywords “public”, “protected” and “private”.
- Can use “static” keyword to declared a static function.

	Inside	Children	Outside
Public	YES	YES	YES
Protected	YES	YES	NO
Private	YES	NO	NO

OBJECT

- An individual instance of the data structure defined by a class.

  Object also known as instance.
- Using “new” keyword to create object from class.
- It’s an instance of a Class.
- Object is specific.

```php
<?php
class Hero
{
    protected $name;

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

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

    public function display()
    {
        echo $this->name;
    }

}

$antimage = new Hero;
$antimage->setName('Antimage');
$antimage->display();
```

INHERITANCE

- It lets subclass inherits characteristics of the parent class.
- Using keyword “extends” to inherit from its parent class.
- All of property and methods will be inherit by its child class, except the one has keyword “private”.

```php
<?php
class Antimage extends Hero
{
    public function __construct()
    {
        parent::__construct("ANTIMAGE");
    }

    public function setName($name)
    {
        echo "You can not setName is $name for this hero.<br>";
        return $this;
    }
}

$nam = new Antimage();
$nam->setName("ABC")->display();
```

POLYMORPHISM

- The provision of a single interface to entities of different types.
- It's able to process objects differently depending on their data type or class.
- Can use "interface" keyword to make interface class, use "implements" to implement an interface.
- Interfaces are skeletons.

```php
<?php
interface Animal
{
    public function name();
}

class Dog implements Animal
{
    public function name()
    {
        echo "This is a Dog";
        return $this;
    }
}

class Cat implements Animal
{
    public function name()
    {
        echo "This is a Cat";
        return $this;
    }
}
```

```php
<?php
class Animal
{
    public function makeNoise()
    {

    }
}

class Dog extends Animal
{
    public function makeNoise()
    {
        echo "woo woo";
        return $this;
    }
}

class Cat extends Animal
{
    public function makeNoise()
    {
        echo "meow meow";
        return $this;
    }
}
```

ENCAPSULATION

- Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.
- Using "private", "protected" and magic methods to set and get its property is better.

```php
<?php
class Person
{
    private $name;
    private $project;

    public function __set($key, $value)
    {
        return property_exists($this, $key) ? $this->$key = $value : $this;
    }

    public function __get($key)
    {
        return property_exists($this, $key) ? $this->$key : $this;
    }
}

$nam = new Person;
$nam->name = "Nam NH";
$nam->project = "News Website";
echo "His name is $nam->name and his project is $nam->project";
```

ABSTRACTION

- Abstraction is the concept of moving the focus from the details and concrete implementation of things, to the types of things.
- Making the programming simpler, more general and more abstract.
- It's like a generalization instead of a specification
- Can use "abstract" keyword to declared abstract class.

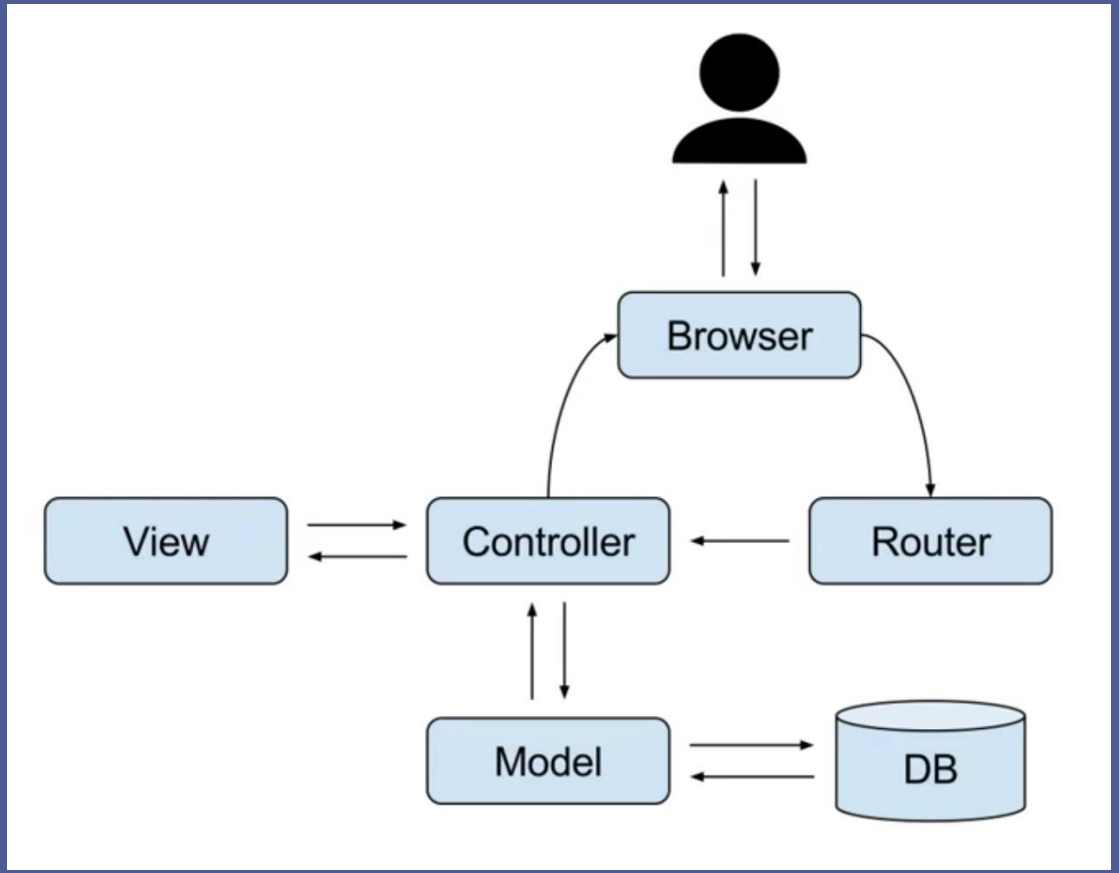
```php
<?php
abstract class Hero
{
    protected $name;
    public function __construct($name = "Undefined")
    {
        $this->name = $name;
    }
    abstract public function setName($name);
    abstract public function getName();
    public function display()
    {
        echo "This one is a hero, his/her name is $this->name";
    }
}

class Antimage extends Hero
{
    public function setName($name)
    {
        $this->name = $name;
        return $this;
    }

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

INTERFACE & ABSTRAC CLASS

- Interface declares what methods a class must have without having to implement them, the one is inherit from abstract class must implement the method its declared as abstract in abstract class.
- Interface is not a class, can not instantiate an interface because it’s do not have any property or method can use immediately. Abstract class is a class, can have normal properties and methods as a normal class, it can be instantiated as a normal class.
- Interface is 100% abstraction, abstract can be or can be not.

MVC ARCHITECTURE

- MVC stand for Model View Controller, is a software design pattern
- Model – The lowest level of the pattern which is responsible for maintaining data.
- View – This is responsible for displaying all or a portion of the data to the user.
- Controller – Software Code that controls the interactions between the Model and View.



TIME TO PRACTICE.