Pentalog

PHP Training

Episode 3

by Andrei Kojusko, 2018

Lesson contents

- Classes
- Objects, inheritance
- Namespaces
- Static methods/variables
- Late static binding
- Some magic methods
- Final classes & methods
- Interfaces / abstract classes
- Traits
- Exceptions, errors & handling
- Autoloading

Classes

```php
class SimpleClass {

    // constant declaration
    public const BAR = 'bar';

    // property declaration
    public $var = 'a default value';

    // method declaration
    public function displayVar(): string {

        return $this->var;

    }
}
```

- public
- protected
- private

- abstract
- final

- $this
- self::
- static::
- parent::

Objects

```php
$instance1 = new SimpleClass();
$instance1->var = 'I\'m a SimpleClass var';

$instance2 = new SimpleClass();

$className = 'SimpleClass';
$instance3 = new $className();

$instance4 = clone $instance1;

$instance5 = $instance1;
```

```php
$instance1 === $instance2 // false

$instance1 === $instance5 // true

$instance1 === $instance4 // false
```

Inheritance

```php
class ExtendClass extends SimpleClass {

    // Redefine the parent method
    public function displayVar(): string
    {
        echo "Extending class\n";
        return parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
```

Only one parent!

Namespaces

```php
<?php

namespace MyProject;

const CONNECT_OK = 1;

class SimpleClass { /* ... */ }

function connect() { /* ... */ }


new SimpleClass();

new \OtherProject\OtherClass();

SimpleClass::class // MyProject\SimpleClass
```

```php
$instance = new MyProject\ExtendClass();

$instance::class // MyProject\ExtendClass


namespace MyProject;


$instance::class === SimpleClass::class // false

$instance instanceof SimpleClass // true
```

Static methods/variables

```php
class Foo {

    public static $myStaticVar = 'foo';

    public static function aStaticMethod() {
        // ...
    }

}


Foo::aStaticMethod();

Foo::$myStaticVar
```

Constants are static

Late static binding

```php
class A {
    public static function who() {
        echo __CLASS__;
    }

    public static function test() {
        self::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test(); // A
```

```php
class A {
    public static function who() {
        echo __CLASS__;
    }

    public static function test() {
        static::who();
    }
}

class B extends A {
    public static function who() {
        echo __CLASS__;
    }
}

B::test(); // B
```

Some magic methods

- __construct(...$args)
- __clone()
- __destruct()
- __toString()
- __get(string $name)
- __set(string $name, $value)
- __invoke(...$args)
- __call(string $name, array $arguments)
- __callStatic(string $name, array $arguments)
- JsonSerializable::jsonSerialize

Final classes & methods

```
class BaseClass {
   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo "ChildClass::moreTesting() called\n";
   }
}

// Fatal error: Cannot override final method
BaseClass::moreTesting()
```

```
final class BaseClass {

}

class ChildClass extends BaseClass {

}

// Fatal error: Class ChildClass may not inherit from final
class (BaseClass)
```

Abstract class

```
abstract class AbstractClass {

    // Force Extending class to define this method
    abstract protected function getValue();

    // Common method
    public function printOut() {

        print $this->getValue() . "\n";

    }

}
```

```
class ConcreteClass1 extends AbstractClass {

    protected function getValue() {

        return "ConcreteClass1";

    }

}
```

Interfaces

```php
interface Timestampable {

    public function setTimestamp(int $timestamp): void;
    public function getTimestamp(): ?int;

}

interface Blameable {

    public function setBlame(UserInterface $user): void;

}
```

```php
class Log implements Timestampable, Blameable {

    public function setTimestamp(int $timestamp): void
    {
        $this->timestamp = $timestamp;
    }

    public function setBlame(UserInterface $user): void
    {
        $this->blameName = $user->getName();
    }

    public function getTimestamp(): ?int
    {
        return $this->timestamp;
    }

}
```

Traits

```php
trait Hello {
    public function sayHello(): void {
        echo 'Hello ';
    }
}
```

```php
trait World {
    public function sayWorld(): void {
        echo 'World';
    }
}
```

```php
class MyHelloWorld {

    use Hello, World;

    public function sayExclamationMark(): void {

        echo '!';

    }

}

$o = new MyHelloWorld();
$o->sayHello();
$o->sayWorld();
$o->sayExclamationMark();
```

Traits - overriding

```php
trait HelloWorld {
    public function sayHello(): void {
        echo 'Hello World!';
    }
}
```

```php
class MyClass {
    use HelloWorld { sayHello as private myPrivateHello; }
}
```

Exceptions, errors & handling

```php
function inverse($x) {
    if (!$x) {
        throw new \Exception('Division by zero.');
    }
    return 1/$x;
}
```

```php
try {
    echo inverse(0);
} catch (\Exception | \Error $e) {
    echo 'Caught exception: '.$e->getMessage();
} finally {
    echo "Finally.";
}
```

Autoloading

```php
spl_autoload_register(function ($class) {
    $file = 'src'.DIRECTORY_SEPARATOR.str_replace('\\', DIRECTORY_SEPARATOR, $class).'.php';
    require $file;
});
```

Links

http://be2.php.net/manual/en/language.oop5.php

http://be2.php.net/manual/en/language.namespaces.php

http://php.net/manual/en/language.oop5.magic.php

http://php.net/manual/en/language.oop5.traits.php

http://be2.php.net/manual/en/language.errors.php7.php

http://be2.php.net/manual/en/language.exceptions.php

Homework

No homework :) But please, read the docs.