PHP Objects

Dr. Charles Severance
www.wa4e.com

http://www.wa4e.com/code/objects.zip

CC BY

PHP => Object Oriented

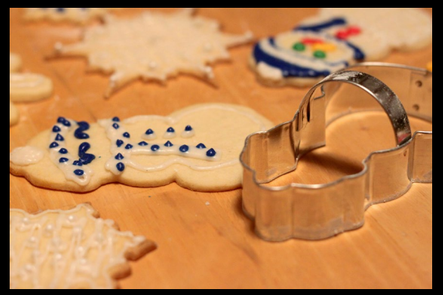
- With PHP 5, an object oriented approach is the preferred pattern.
- Libraries are evolving toward OOP.

Object Oriented Programming (OOP)

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

http://en.wikipedia.org/wiki/Object-oriented_programming

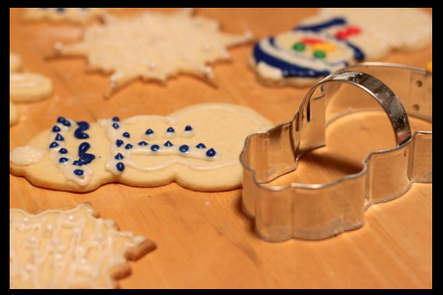
Definitions



- Class - a template - Dog
- Method - A defined capability of a class - bark()
- Object or Instance - A particular instance of a class - Lassie

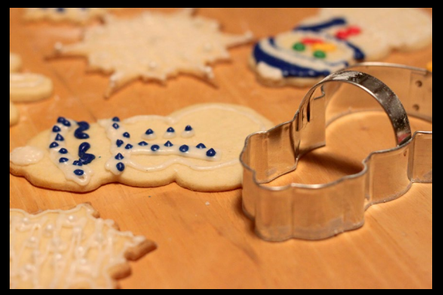
Image CC-By 2.0: https://www.flickr.com/photos/dinnerseries/23570475099

Terminology: Class



Defines the abstract characteristics of a thing (object), including the thing's characteristics (its attributes, fields, or properties) and the thing's behaviors (the things it can do, or methods, operations, or features). One might say that a class is a blueprint or factory that describes the nature of something. For example, the class Dog would consist of traits shared by all dogs, such as breed and fur color (characteristics), and the ability to bark and sit (behaviors).

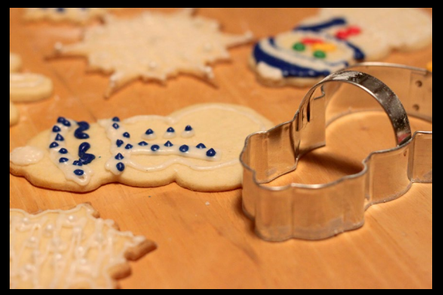
Terminology: Instance



One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the Lassie object is an instance of the Dog class. The set of values of the attributes of a particular object is called its state. The object consists of state and the behavior that is defined in the object's class.

Object and Instance are often used interchangeably.

Terminology: Method



An object's abilities. In language, methods are verbs. Lassie, being a Dog, has the ability to bark. So bark() is one of Lassie's methods. She may have other methods as well, for example sit() or eat() or walk() or save_timmy(). Within the program, using a method usually affects only one particular object; all Dogs can bark, but you need only one particular dog to do the barking.

Method and Message are often used interchangeably.

Non-OOP

Date/Time Functions

Cha

Table of Contents

http://php.net/manual/en/ref.datetime.php

OOP

The DateTime class

Change language: English

(PHP 5 >= 5.2.0)

Introduction

Representation of date and time.

Class synopsis

```
DateTime implements DateTimeInterface {

    /* Constants */
    const string ATOM = "Y-m-d\TH:i:sP" ;
    const string COOKIE = "l, d-M-y H:i:s T" ;
    const string ISO8601 = "Y-m-d\TH:i:sO" ;
    const string RFC822 = "D, d M y H:i:s O" ;
    const string RFC850 = "l, d-M-y H:i:s T" ;
    const string RFC1036 = "D, d M y H:i:s O" ;
    const string RFC1123 = "D, d M Y H:i:s O" ;
    const string RFC2822 = "D, d M Y H:i:s O" ;
    const string RFC3339 = "Y-m-d\TH:i:sP" ;
    const string RSS = "D, d M Y H:i:s O" ;
    const string W3C = "Y-m-d\TH:i:sP" ;

    /* Methods */
    public __construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )
    public DateTime add ( DateInterval $interval )
    public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone
    $timezone ] )
    public static array getLastErrors ( void )
    public DateTime modify ( string $modify )
```

http://www.php.net/manual/en/class.datetime.php

```php
<?php
date_default_timezone_set('America/New_York');

$nextWeek = time() + (7 * 24 * 60 * 60);
echo 'Now:       '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";

echo("=====\n");

$now = new DateTime();
$nextWeek = new DateTime('today +1 week');
echo 'Now:       '. $now->format('Y-m-d') ."\n";
echo 'Next Week: '. $nextWeek->format('Y-m-d') ."\n";
```

```
Now:       2013-09-25
Next Week: 2013-10-02
=====
Now:       2013-09-25
Next Week: 2013-10-02
```

date.php

Creating Objects in PHP

nonobj.php

```
<?php
$chuck = array("fullname" => "Chuck Severance",
  'room' => '4429NQ');
$colleen = array("familyname" => "van Lent",
  'givenname' => 'Colleen', 'room' => '3439NQ');

function get_person_name($person) {
     if ( isset($person['fullname']) ) return $person['fullname'];
     if ( isset($person['familyname']) && isset($person['givenname']) ) {
          return $person['givenname'] . ' ' . $person['familyname'] ;
     }
     return false;
}

print get_person_name($chuck) . "\n";
print get_person_name($colleen) . "\n";
```

```
Chuck Severance
Colleen van Lent
```

withobj.php

```php
<?php
class Person {
    public $fullname = false;
    public $givenname = false;
    public $familyname = false;
    public $room = false;
    function get_name() {
        if ( $this->fullname !== false ) return $this->fullname;
        if ( $this->familyname !== false && $this->givenname !== false ) {
            return $this->givenname . ' ' . $this->familyname;
        }
        return false;
    }
}

$chuck = new Person();
$chuck->fullname = "Chuck Severance";
$chuck->room = "4429NQ";

$colleen = new Person();
$colleen->familyname = 'van Lent';
$colleen->givenname = 'Colleen';
$colleen->room = '3439NQ';

print $chuck->get_name() . "\n";
print $colleen->get_name() . "\n";
```

```
Chuck Severance
Colleen van Lent
```

Object Oriented Libraries

Two New Operators

- Access "static item" in a class

```
echo DateTime::RFC822."\n";
```

- Access item in an object

```
echo $z->format('Y-m-d')."\n";
```

```php
DateTime implements DateTimeInterface {

    /* Constants */
    const string ATOM = "Y-m-d\TH:i:sP" ;
    const string COOKIE = "l, d-M-y H:i:s T" ;
    const string ISO8601 = "Y-m-d\TH:i:sO" ;
    const string RFC822 = "D, d M y H:i:s O" ;
    const string RFC850 = "l, d-M-y H:i:s T" ;
    const string RFC1036 = "D, d M y H:i:s O" ;
    const string RFC1123 = "D, d M Y H:i:s O" ;
    const string RFC2822 = "D, d M Y H:i:s O" ;
    const string RFC3339 = "Y-m-d\TH:i:sP" ;
    const string RSS = "D, d M Y H:i:s O" ;
    const string W3C = "Y-m-d\TH:i:sP" ;

    /* Methods */
    public __construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )
    public DateTime add ( DateInterval $interval )
    public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone
    $timezone ] )
    public static array getLastErrors ( void )
    public DateTime modify ( string $modify )
    public static DateTime __set_state ( array $array )
    public DateTime setDate ( int $year , int $month , int $day )
    public DateTime setISODate ( int $year , int $week [, int $day = 1 ] )
    public DateTime setTime ( int $hour , int $minute [, int $second = 0 ] )
    public DateTime setTimestamp ( int $unixtimestamp )
```

```php
echo DateTime::RFC822."\n";
```

```
D, d M y H:i:s O
```

```php
DateTime implements DateTimeInterface {

    /* Constants */
    const string ATOM = "Y-m-d\TH:i:sP" ;
    const string COOKIE = "l, d-M-y H:i:s T" ;
    const string ISO8601 = "Y-m-d\TH:i:sO" ;
    const string RFC822 = "D, d M y H:i:s O" ;
    const string RFC850 = "l, d-M-y H:i:s T" ;
    const string RFC1036 = "D, d M y H:i:s O" ;
    const string RFC1123 = "D, d M Y H:i:s O" ;
    const string RFC2822 = "D, d M Y H:i:s O" ;
    const string RFC3339 = "Y-m-d\TH:i:sP" ;
    const string RSS = "D, d M Y H:i:s O" ;
    const string W3C = "Y-m-d\TH:i:sP" ;

    /* Methods */
    public __construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )
    public DateTime add ( DateInterval $interval )
    public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone
    $timezone ] )
    public static array getLastErrors ( void )
    public DateTime modify ( string $modify )
    public static DateTime __set_state ( array $array )
    public DateTime setDate ( int $year , int $month , int $day )
```

```php
$x = new DateTime();
$y = new DateTime('now');
$z = new DateTime('2012-01-31');
```

```
DateTime implements DateTimeInterface {

    /* Constants */
    const string ATOM = "Y-m-d\TH:i:sP" ;
    const string COOKIE = "l, d-M-y H:i:s T"
    const string ISO8601 = "Y-m-d\TH:i:sO" ;
    const string RFC822 = "D, d M y H:i:s O"
    const string RFC850 = "l, d-M-y H:i:s T"
    const string RFC1036 = "D, d M y H:i:s O
    const string RFC1123 = "D, d M Y H:i:s O
    const string RFC2822 = "D, d M Y H:i:s O
    const string RFC3339 = "Y-m-d\TH:i:sP" ;
    const string RSS = "D, d M Y H:i:s O" ;
    const string W3C = "Y-m-d\TH:i:sP" ;

    /* Methods */
    public __construct ([ string $time = "now
    public DateTime add ( DateInterval $inter
    public static DateTime createFromFormat (
    $timezone ] )
    public static array getLastErrors ( void
    public DateTime modify ( string $modify )
    public static DateTime __set_state ( array $array )
    public DateTime setDate ( int $year , int $month , int $day )
```

```
$x = new DateTime('1999-04-31');
$oops = DateTime::getLastErrors();
print_r($oops);


Array
(
    [warning_count] => 1
    [warnings] => Array
        (
            [11] => The parsed date was invalid
        )
    [error_count] => 0
    [errors] => Array
        (
        )
)
```

```
    const string W3C = "Y-m-d\TH:i:sP" ;

    /* Methods */
    public __construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )
    public DateTime add ( DateInterval $interval )
    public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone
    $timezone ] )
    public static array getLastErrors ( void )
    public DateTime modify ( string $modify )
    public static DateTime __set_state ( array $array )
    public DateTime setDate ( int $year , int $month , int $day )
    public DateTime setISODate ( int $year , int $week [, int $day = 1 ] )
    public DateTime setTime ( int $hour , int $minute [, int $second = 0 ] )
    public DateTime setTimestamp ( int $unixtimestamp )
    public DateTime setTimezone ( DateTimeZone $timezone )
    public DateTime sub ( DateInterval $interval )
    public DateInterval diff ( DateTimeInterface $datetime2 [
    public string format ( string $format )
    public int getOffset ( void )
    public int getTimestamp ( void )
    public DateTimeZone getTimezone ( void )
    public __wakeup ( void )
}
```

```
$z = new DateTime('2012-01-31');
echo $z->format('Y-m-d')."\n";
```

```
2012-01-31
```

```
    public __construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )
    public DateTime add ( DateInterval $interval )
    public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone
    $timezone ] )
    public static array getLastErrors ( void )
    public DateTime modify ( string $modify )
    public static DateTime __set_state ( array $array )
    public DateTime setDate ( int $year , int $month , int $day )
    public DateTime setISODate ( int $year , int $week [, int $day = 1 ] )
    public DateTime setTime ( int $hour , int $minute [, int $second = 0 ] )
    public DateTime setTimestamp ( int $unixtimestamp )
    public DateTime setTimezone ( DateTimeZone $timezone )
    public DateTime sub ( DateInterval $interval )
    public DateInterval diff ( DateTimeInterface $datetime2 [, bool $absolute = false ] )
    public string format ( string $format )
    public int getOffset ( void )
    public int getTimestamp ( void )
    public DateTimeZone getTimezone ( void )
    public __wakeup ( void )
}
```

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

Object Life Cycle in PHP

http://en.wikipedia.org/wiki/Constructor_(computer_science)

Object Life Cycle

- Objects are created, used and discarded.
- We have special blocks of code (methods) that get called
  - At the moment of creation (constructor)
  - At the moment of destruction (destructor)
- Constructors are used a lot.
- Destructors are seldom used.

Constructor

The primary purpose of the constructor is to set up some instance variables to have the proper initial values when the object is created.

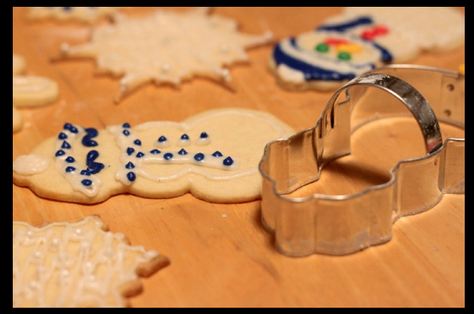
party.php

```
class PartyAnimal {
    function __construct() {
        echo("Constructed\n");
    }
    function something() {
        echo("Something\n");
    }
    function __destruct() {
        echo("Destructed\n");
    }
}

echo("--One\n");
$x = new PartyAnimal();
echo("--Two\n");
$y = new PartyAnimal();
echo("--Three\n");
$x->something();
echo("--The End?\n");
```

```
--One
Constructed
--Two
Constructed
--Three
Something
--The End?
Destructed
Destructed
```

Many Instances



- We can create lots of objects - the class is the template for the object.
- We can store each distinct object in its own variable.
- We call this having multiple instances of the same class.
- Each instance has its own copy of the instance variables.

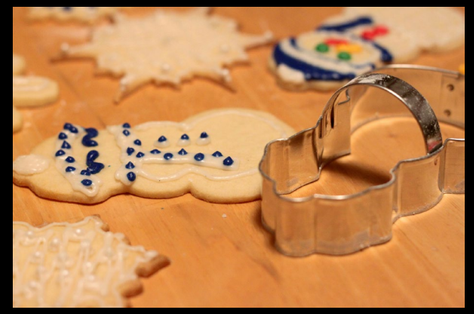
hello.php

```
class Hello {
    protected $lang; // Only accessible in the class
    function __construct($lang) {
        $this->lang = $lang;
    }
    function greet() {
        if ( $this->lang == 'fr' ) return 'Bonjour';
        if ( $this->lang == 'es' ) return 'Hola';
        return 'Hello';
    }
}

$hi = new Hello('es');
echo $hi->greet()."\n";
```

```
Hola
```

Definitions



- Class - a template - Dog
- Method or Message - A defined capability of a class - bark()
- Object or Instance - A particular instance of a class - Lassie
- Constructor - A method which is called when the instance / object is created

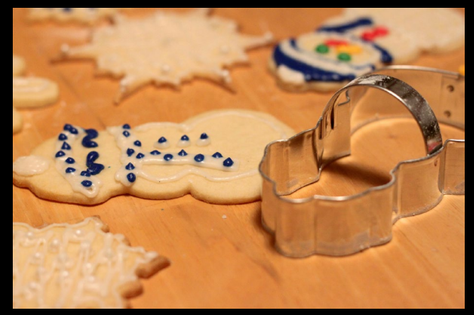
Object Inheritance in PHP

http://www.php.net/manual/en/language.oop5.inheritance.php

Inheritance

- When we make a new class we can reuse an existing class and inherit all the capabilities of an existing class and then add our own little bit to make our new class
- Another form of store and reuse
- Write once - reuse many times
- The new class (child) has all the capabilities of the old class (parent) - and then some more

Terminology: Inheritance



"Subclasses" are more specialized versions of a class, which inherit attributes and behaviors from their parent classes, and can introduce their own.

http://en.wikipedia.org/wiki/Object-oriented_programming

goodbye.php

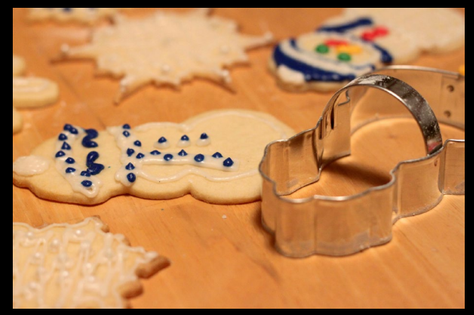
```
class Hello {
    protected $lang;
    function __construct($lang) { ... }
    function greet() { ... }
}

class Social extends Hello {
    function bye() {
     if ( $this->lang == 'fr' ) return 'Au revoir';
     if ( $this->lang == 'es' ) return 'Adios';
     return 'goodbye';
    }
}

$hi = new Social('es');
echo $hi->greet()."\n";
echo $hi->bye()."\n";
```

```
Hola
Adios
```

Definitions



- Class - a template - Dog
- Method or Message - A defined capability of a class - bark()
- Object or Instance - A particular instance of a class - Lassie
- Constructor - A method which is called when the instance / object is created
- Inheritance - the ability to take a class and extend it to make a new class

Visibility

Class member variables also have scope.

- Public – can be accessed outside the class, inside the class, and in derived classes
- Protected – can be accessed inside the class, and in derived classes
- Private – can only be accessed inside the class (i.e. private variables are not visible in derived classes)

http://www.php.net/manual/en/language.oop5.visibility.php

visibility.php

```
class MyClass
{
    public $pub = 'Public';
    protected $pro = 'Protected';
    private $priv = 'Private';

    function printHello()
    {
        echo $this->pub."\n";
        echo $this->pro."\n";
        echo $this->priv."\n";
    }
}

$obj = new MyClass();
echo $obj->pub."\n"; // Works
echo $obj->pro."\n"; // Fatal Error
echo $obj->priv."\n"; // Fatal Error
$obj->printHello(); // Shows Public, Protected and Private
```

```
Public
Public
Protected
Private
```

```
class MyClass2 extends MyClass
{

    function printHello()
    {
        echo $this->pub."\n";
        echo $this->pro."\n";
        echo $this->priv."\n"; // Undefined
    }
}

echo("--- MyClass2 ---\n");
$obj2 = new MyClass2();
echo $obj2->pub."\n"; // Works
$obj2->printHello(); // Shows Public, Protected, Undefined
```

```
--- MyClass2 ---
Public
Public
Protected
(false)
```

Building an Object from Scratch

- Sometimes a developer will prefer to make an object with public key-value pairs rather than an array.
- Use where appropriate…

```php
$player = new stdClass();

$player->name = "Chuck";
$player->score = 0;

$player->score++;

print_r($player);

class Player {
    public $name = "Sally";
    public $score = 0;
}

$p2 = new Player();
$p2->score++;

print_r($p2);
```

scratch.php

```
stdClass Object
(
    [name] => Chuck
    [score] => 1
)
Player Object
(
    [name] => Sally
    [score] => 1
)
```

Summary

- Object Oriented programming is a very structured approach to code reuse.
- There is a trend away from global function names and toward OO.
- We can group data and functionality together and create many independent instances of a class.

Acknowledgements / Contributions

These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) as part of www.wa4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here

Additional Source Information

- Snowman Cookie Cutter" by Didriks is licensed under CC BY
  https://www.flickr.com/photos/dinnerseries/23570475099
- Photo from the television program *Lassie*. Lassie watches as Jeff (Tommy Rettig) works on his bike is Public Domain
  https://en.wikipedia.org/wiki/Lassie#/media/File:Lassie_and_Tommy_Rettig_1956.JPG