



Introduzione OOP PHP

Slide tratte da:

W3CSCHOOLS.COM

(https://www.w3schools.com/php/php_oop_what_is.asp)



Introduzione all'OOP in PHP

- **PHP** implementa il **paradigma** della Programmazione Orientata agli Oggetti
- **OOP** sta per Object-Oriented Programming
- Impareremo **come** il PHP **implementa i concetti di base** della programmazione orientata agli oggetti.



Vantaggi dell'OOP

- Vantaggi dell'OOP rispetto alla programmazione procedurale
- Offre numerosi vantaggi rispetto alla programmazione procedurale, tra cui:
 - Velocità ed esecuzione semplificata.
 - Struttura più chiara del codice.
 - Rispetto del principio "DRY" (*Don't Repeat Yourself*).
 - Migliore manutenibilità del codice
 - Possibilità di creare applicazioni riutilizzabili ottimizzando codice e tempi.

Classe e Oggetto

- Cosa sono le **classi** e gli **oggetti**?
- Le classi e gli oggetti sono i due **concetti principali** dell'OOP.
- Una **classe** è un **modello** per gli oggetti,
- Un **oggetto** è un'**istanza** di una classe.

Esempio di Classi e Oggetti

● Consideriamo i seguenti esempi:

- Classe: Fruit
 - Oggetti: *Apple, Banana, Mango*
- Classe: Car
 - Oggetti: *Volvo, Audi, Toyota*

Definizione di una classe

- Abbiamo la seguente sintassi per definire una classe in PHP

PHP

```
class NomeClasse {  
  
    // Proprietà e metodi vanno qui...  
  
}
```

Esempio di codice di una classe

- Definiamo la classe *Fruit* con 2 proprietà e i metodi getter/setter:

PHP

```
class Fruit {  
    public $name;  
    public $color;  
  
    function set_name($name) {  
        $this->name = $name;  
    }  
  
    function get_name() {  
        return $this->name;  
    }  
}
```

Creare oggetti

- Per creare oggetti da una classe, utilizziamo la parola chiave **"new"**.
- Creiamo dalla classe Fruit due oggetti: apple e banana:

PHP

```
$apple = new Fruit();  
$banana = new Fruit();
```

Esempio di utilizzo

- In questo esempio creiamo un oggetto
- e utilizziamo i metodi getter/setter per l'attributo `name`

PHP

```
$apple = new Fruit();  
$apple->set_name('Mela');  
echo $apple->get_name();
```

La parola chiave `$this`

- La parola chiave **`$this`** si riferisce all'**oggetto corrente**
- è disponibile **solo all'interno dei metodi**

La parola chiave `instanceof`

- Puoi utilizzare la `instanceof` per verificare se un oggetto appartiene a una classe specifica

HTML

```
$apple = new Fruit();  
echo $apple instanceof Fruit;
```



Costruttore

- Il costruttore è una funzione speciale chiamata automaticamente alla creazione di un oggetto.
- In PHP il costruttore è chiamato `__construct()`
- Il metodo `__construct()` può accettare dei parametri formali alla chiamata.



Esempio di costruttore

```
PHP

class Fruit {
    public $name;

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

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

$apples = new Fruit("Mela");
```



Distruttore

- Il distruttore è una funzione speciale chiamata automaticamente quando un oggetto viene distrutto.
- Oppure alla fine di uno script
- In PHP il costruttore è chiamato `__destruct()`



Esempio di distruttore

```
PHP

class Fruit {
    public $name;

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

    function __destruct() {
        echo "Il frutto è {$this->name}.";
    }
}

$apples = new Fruit("Mela");
```

Modificatori di accesso

- Proprietà e metodi possono avere tre modificatori di accesso:
 - public,
 - protected
 - private
- Questi controllano dove possono essere accessibili.

Modificatori di accesso

- **public** è possibile accedere alla proprietà o al metodo da qualsiasi luogo. Questa è l'impostazione predefinita
- **protected** è possibile accedere alla proprietà o al metodo all'interno della classe e dalle classi derivate da quella classe
- **private** è possibile accedere alla proprietà o al metodo solo all'interno della classe

Modificatori di accesso - esempio

PHP

```
class Fruit {
    public $name;
    protected $color;
    private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango';      // OK
$mango->color = 'Giallo';    // ERRORE
$mango->weight = '300';      // ERRORE
```

Modificatori di accesso - esempio

PHP

<pre>class Fruit { private \$name; private \$color; private \$weight; function set_name(\$n) { \$this->name = \$n; } protected function set_color(\$n) { \$this->color = \$n; } private function set_weight(\$n) { \$this->weight = \$n; } }</pre>	<pre>\$mango = new Fruit(); \$mango->set_name('Mango'); // OK \$mango->set_color('Yellow'); // ERROR \$mango->set_weight('300'); // ERROR</pre>
--	---

Modificatori di accesso - Nota

- L'uso adeguato dei modificatori di accesso contribuisce alla sicurezza e alla struttura del codice.
- Protegge le informazioni sensibili e definisce chiaramente come i membri di una classe possano essere utilizzati.

Riepilogo

- Abbiamo visto i fondamenti della OOP in PHP
- Concetti come classi, oggetti, costruttori, distruttori e modificatori di accesso.
- Questi concetti saranno fondamentali per la scrittura di codice PHP più organizzato e manutenibile.

Abbiamo finito!

Ora è importante fare parecchi esercizi!

