Università Degli Studi di Ferrara

Corso di Laurea in Informatica - A.A. 2023 - 2024

Tecnologie Web

Lez. 05 - OOPHP

Tecnologie Web - Docente: Giacomo Piva - giacomo.piva@unife.it

Cap. 1

Tecnologie Server Side

In questa lezione…

- OOPHP
- PHP e HTML mixing
- PHP 7

Personal **H**ome **P**age Tool

PHP **H**ypertext **P**reprocessor

https://www.php.net

1994 Rasmus Lerdorf

Il linguaggio PHP

Sintassi simile al linguaggio **C**

Largamente utilizzato (Wordpress, Joomla, Magento …)

Ottime prestazioni (specialmente dalla versione 7.0)

Il linguaggio PHP

Linguaggio interpretato

Linguaggio Object Oriented (dalla versione 3.0)

Linguaggio a tipizzazione dinamica (dynamic typing)

Il linguaggio PHP

Inserire il codice PHP in una pagina HTML è molto semplice:

- Nominare il file come `.php` e non `.html`
- Inserire il codice PHP fra i tag `<?php` e `?>`

Ad esempio:

```
<h1><?php echo “Titolo”; ?></h1>
```

Il linguaggio PHP

Altri semplici esempi:

```
<?php $titolo = “Titolo”; ?>
<h1> <?php echo $titolo; ?> </h1>

<?php $prezzo = 5.00 + (5.00*0.22); ?>
<p> <?php echo $prezzo . ”€”; ?> </p>
```

Esempio 1

Codice PHP

Classi

Le classi sono l’entità base della programmazione ad oggetti.

Ci consentono di definire dei tipi di dato “complessi” attraverso cui possiamo rappresentare il mondo reale e descriverne caratteristiche e comportamento.

Possiamo pensare alle classi come a delle struct con i superpoteri.

Classi

In C possiamo definire le struct con:

```
struct Person {
  char *first_name;
  char *last_name;
}
```

Classi

In PHP usando la programmazione ad oggetti diventa:

```
class Person {
  public $first_name;
  public $last_name;
}
```

Classi

Quelli che vengono chiamati “campi” in C, nella programmazione ad oggetti vengono chiamate proprietà (property)

Per definire una nuova variabile di tipo “Person” usiamo la keyword `new`

```
$p = new Person();
```

Nella programmazione ad oggetti, la variabile costruita da una classe, prende il nome di **Oggetto** o **Istanza** della **Classe**

Proprietà

Per accedere alle proprietà di un oggetto si utilizza l'operatore freccia `->` ad esempio:

```
echo $p->first_name;
```

Metodi

Le classi possono definire non solo delle proprietà, ma anche dei **metodi**, cioè delle funzioni che agiscono sull'oggetto della classe.

`$this` è la variabile che all'interno della nostra classe fa riferimento all'oggetto stesso.

Esempio 2

Proprietà e Metodi

Costruttore

Assegnare il valore di una proprietà dall’esterno della nostra classe ci è utile (se non necessario) ma non è consigliabile:

```
$p = new Person();

$p->first_name = “John”;
```

Costruttore

Per assegnare il valore di una proprietà ci sono diversi modi, ad esempio per assegnare il valore iniziale è necessario utilizzare un metodo speciale che si chiama **metodo Costruttore**

Il compito del metodo costruttore è quello di inizializzare le proprietà in modo **corretto** e **controllato**.

Esempio 3

Metodo Costruttore

Metodi Getter e Setter

Per accedere correttamente al valore di una proprietà (assegnare e ottenere il valore) è bene passare per un metodo che ne controlla la corretta valorizzazione.

Metodo **Get**ter => Ottenere il valore della property

Metodo **Set**ter => Assegnare il valore alla property

Esempio 4

Metodi Getter e Setter

Modificatore Public

Non c’è nessuna restrizione di accesso ad una proprietà o un metodo marcato come **public**.

Modificatore Private

Ad una proprietà o un metodo marcato come **private** possono accedere tutti i metodi che sono definiti all’interno della stessa classe.

Accesso alle property

Regola Generale

Property: **private**

Metodi: **public**

Object Oriented Programming

3 concetti chiave:

- Incapsulamento (Encapsulation)
- Ereditarietà (Inheritance)
- Polimorfismo (Polymorphism)

OOP Incapsulamento

Ci consente di organizzare e raggruppare i dati e di controllare l’accesso ai dati.

OOP Incapsulamento

È ciò che abbiamo visto fin ora.

La stesura di una Classe è un esempio di incapsulamento (raggruppare i dati).

L’utilizzo dei modificatori di accesso (public, private …) sono un esempio di controllo di accesso ai dati.

Esercizio 1

Utilizzando la classe `Person` vista nelle slides precedenti, scrivete una pagina HTML contenente un **form** per l'inserimento dei dati richiesti (nome e cognome).

L'invio del form passa i dati via POST ad una **nuova pagina** che li utilizzerà prima per istanziare un oggetto attraverso il **costruttore** e poi, attraverso i **metodi getter**, per visualizzare l'oggetto.

Includete un file css per dare un po' di stile alla pagina.

OOP Ereditarietà

Ci consente di creare legami di gerarchia fra i dati come succede fra padre e figlio.

È un modo per struttura e riutilizzare il codice applicando la direttiva “chiave” della buona programmazione D. R. Y.

D	R	Y
Don’t	Repeat	Yourself

OOP Ereditarietà

Cambiamo esempio e consideriamo un punto.

Per una classe che descrive un punto possiamo immaginare:

- Coordinata X
- Coordinata Y
- Set delle coordinate
- Get delle coordinate

Esempio 5

La classe Point

OOP Ereditarietà

Pensiamo ora ad una ipotetica classe Pixel

Potremmo immaginare di aggiungere:

- Colore

Creiamo quindi una nuova classe (Pixel) figlia della classe Point che ne estende le funzioni.

Esempio 6

La classe Pixel estende Point

Esercizio 2

Utilizzando la classe `Point` vista nelle slides precedenti, scrivete una pagina HTML/PHP che definisce la classe stessa e la sua classe figlia `Pixel`.

La pagina deve contenere un **form** per l'inserimento delle **coordinate** e del **colore**.

L'invio del form passa i dati via POST alla **pagina stessa** che li utilizzerà prima per istanziare un `pixel` attraverso il **costruttore** e poi, attraverso i **metodi setter** aggiornare le coordinate ed infine visualizzare coordinate e colore.

Suggerimento: Inserite dei controlli per non avere problemi al primo caricamento della pagina.

Includete un file css per dare un po' di stile alla pagina.

Modificatore Protected

Ad una proprietà o un metodo marcato come **protected** possono accedere tutti i metodi che sono definiti all'interno della classe o all'interno di sottoclassi.

Modificatore Static

Il modificatore **static** segnala che il valore della property non dipende dall'oggetto, ma per tutte le istanze di una stessa classe, assume lo stesso valore.

Property Static

```php
class Auto {

    private $posti;

    private $porte;

    static public $ruote = 4;

}

echo Auto::$ruote;
```

Modificatore Static

Il modificatore **static** segnala che il metodo non agisce sull'istanza ma sulla Classe.

Un esempio, è il metodo costruttore, che a rigor di logica **potrebbe essere** definito come:

```
static public function __construct();
```

Ma, si sa, il metodo costruttore è un metodo speciale.

Metodi Static

```
class MyClass {
    static public function myMethod(){ … }
}

echo MyClass::myMethod();
```

Esempio 7

Property e Metodi Statici

OOP Polimorfismo

Ci consente di dare più implementazioni ad una stessa funzionalità (**overriding** dei metodi).

Ci consente di utilizzare le funzionalità della superclasse su oggetti istanze della sottoclasse.

Esempio 8

Polimorfismo e overriding dei metodi

PHP e HTML mixing

Esiste un modo corretto e un modo sbagliato di “mescolare” insieme codice PHP e HTML:

Questo:

```php
<?php
   $titolo = “Titolo”;
   echo “<h1>”.$titolo.”</h1>”;
?>
```

PHP e HTML mixing

Esiste un modo corretto e un modo sbagliato di “mescolare” insieme codice PHP e HTML:

Questo:

```
<?php
   $titolo = “Titolo”;
   echo “<h1>”.$titolo.”</h1>”;
?>
```

SBAGLIATO!

PHP e HTML mixing

```
if  ($result->num_rows==0)
  {
       echo "Nessun utente trovato";
  }
 else
       {
         echo '<table>';
         echo '<tr>';
         echo '<th>Matricola</th>';
         echo '<th>Nome</th>';
         echo '<th>Cognome</th>';
         echo '</tr>';
         echo '';
```

PHP e HTML mixing

```
if  ($result->num_rows==0)
  {
       echo "Nessun utente trovato";
  }
 else
{
  echo '<table>';
  echo '<tr>';
  echo '<th>Matricola</th>';
  echo '<th>Nome</th>';
  echo '<th>Cognome</th>';
  echo '</tr>';
  echo '';
```

PHP e HTML mixing

Con questa riga:

```
<h1 class=“title” id=“page-title”> … </h1>
```

Quale “tecnica” usereste ?

```
echo “<h1 class=‘title’ id=‘page-title’>”.$titolo …
```

```
echo “<h1 class=\“title\” id=\“page-title\”>”.$titolo …
```

PHP e HTML mixing

Modo corretto:

```
<?php $titolo = “Titolo”; ?>

<h1> <?php echo $titolo; ?> </h1>
```

```php
<div id="primary" class="content-area">
    <main id="main" class="site-main" role="main">

    <?php if ( have_posts() ) : ?>

        <?php if ( is_home() && ! is_front_page() ) : ?>
            <header>
                <h1 class="page-title screen-reader-text"><?php single_post_title(); ?></h1>
            </header>
        <?php endif; ?>
```

Wordpress 4.6 (index.php)

```php
    <div class="skip-links">
        <a href="#header-nav" class="skip-link skip-nav">
            <span class="icon"></span>
            <span class="label"><?php echo $this->__('Menu'); ?></span>
        </a>

        <a href="#header-search" class="skip-link skip-search">
            <span class="icon"></span>
            <span class="label"><?php echo $this->__('Search'); ?></span>
        </a>
```

Magento 1.9 (head.phtml)

```php
<div id="node-<?php print $node->nid; ?>" class="<?php print $classes; ?> clearfix"<?php print $attributes; ?>>

  <?php print $user_picture; ?>

  <?php print render($title_prefix); ?>
  <?php if (!$page): ?>
    <h2<?php print $title_attributes; ?>><a href="<?php print $node_url; ?>"><?php print $title; ?></a></h2>
  <?php endif; ?>
  <?php print render($title_suffix); ?>

  <?php if ($display_submitted): ?>
    <div class="submitted">
      <?php print $submitted; ?>
    </div>
  <?php endif; ?>
```

Drupal 7.9 (node.tpl.php)

PHP e HTML mixing

Imparate al più presto ad usare le forme:

```
<ul>
<?php for( … ) : ?>
 <li> … </li>
<?php endfor; ?>
</ul>
```

```
<?php if( … ) : ?>
 <p> … </p>
<?php else: ?>
 <p> … </p>
<?php endif; ?>
```

Esempi 9 e 10

Array

Esercizio 3

Realizzate uno script PHP che riempia una tabella con il contenuto di un array associativo (multidimensionale).

Includete un file css che dia un po’ di colore alla tabella.

PHP 7

	PHP5		PHP7	
CONCURRENCY	RESPONSE TIME (SEC)	TRANSACTIONS PER SECOND	RESPONSE TIME (SEC)	TRANSACTIONS PER SECOND
5	0.130	39.212	0.060	81.136
10	0.262	37.794	0.130	76.620
20	0.526	37.882	0.258	77.676
40	1.058	37.630	0.520	76.992
80	2.116	37.446	1.046	76.124

PHP 7

- Nuovo Operatore Spaceship
- Null Coalesce (“fondersi” è molto brutto)
- Dichiarazione del tipo di ritorno
- Dichiarazione del tipo per le variabili

Operatore Spaceship

```
$a <=> $b
```

Ritorna nell’ordine:

-1 se `$a` è `<` `$b`
0 se `$a` è `==` `$b`
+1 se `$a` è `>` `$b`

Operatore ??

```
$a ?? $b
```

Avrete (quasi) sicuramente usato:

```php
$name = isset($_POST['nome']) ? $_POST['nome'] : '';
```

```php
$name = $_POST['nome'] ?? '';
```

Dichiarazione del tipo di ritorno

```
function getUser()
{

  return new User;

}
```

Nel primo caso, non esiste nessuna “protezione” sul tipo del valore di ritorno, e potrei eseguire il return di qualsiasi cosa, una stringa, un array, un boolean.

```
function getUser() : User
{

  return new User;

}
```

Utilizzando la dichiarazione del tipo di ritorno della funzione, nel caso eseguissi il return di qualcosa di diverso dal tipo indicato otterrei un errore.

Dichiarazione del tipo per le variabili

```
function setName($name)
{

}
```

Come nell'esempio precedente, in questo caso, non esiste nessuna "protezione" sul tipo del valore del mio parametro $name.

```
function setName(string $name)
{

}
```

Utilizzando la dichiarazione del tipo del parametro, nel caso passassi come parametro una variabile del tipo sbagliato otterrei un errore.

Dichiarazione del tipo per le variabili

Tuttavia esistono 2 metodi per l'utilizzo della dichiarazione del tipo dei parametri:

- `default`
- `strict`

Il metodo default, "tenta" di convertire il valore passato al tipo richiesto.

Prendiamo ad esempio questa dichiarazione:

```
function getTotal(float $a, float $b) { .. }
```

Dichiarazione del tipo per le variabili

Se chiamassi la mia funzione in questo modo:

```
getTotal(2, "1 pera”);
```

Otterrei come risultato che la stringa viene trasformata nel numero intero 1 e poi nel valore float 1.0 e non ci sarebbe nessun errore …

Dichiarazione del tipo per le variabili

Aggiungendo la dichiarazione (in cima al file PHP) per il modo strict:

```
declare(strict_types=1);
```

Se chiamassi la stessa funzione con gli stessi parametri:

```
getTotal(2, "1 pera”);
```

Otterrei come risultato un errore.

N.B. Non ha nessun effetto sulla conversione da `int` a `float` .

Funzioni di ordine superiore

I moderni linguaggi di programmazione fanno largo uso delle funzioni anonime.

Una funzione anonima (spesso chiamata anche **closure**) è una funzione che non ha nome, ha solo parametri e codice.

Le funzioni anonime vengono utilizzate per essere passate ad oggetti, variabili o altre funzioni.

Funzioni anonime

Molti linguaggi consentono questa sintassi, largamente utilizzata per realizzare delle **callback** ovvero passare dei blocchi di codice da eseguire al termine delle operazioni di una funzione.

```
$pippo = function() { … }

object->method( $pippo );

Class::method( $pippo );
```

Esempi 11, 12 e 13

Funzioni di ordine superiore

PHP 8 argomenti

La versione 8 del linguaggio PHP consente di fornire un nome agli argomenti come è possibile su altri linguaggi come Python.

```
$person = new Person(
    $name,
    age: $age,
    email: $email,
);
```

PHP 8 argomenti

Posso poi utilizzare gli argomenti nell’ordine che voglio.

```
$person = new Person(
    age: 10,
    ‘Paolo’,
    email: ‘paolo@email.com’,
);
```

PHP 7 vs PHP 8

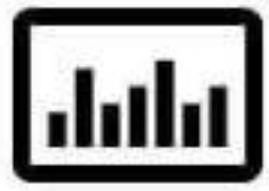

Fundraiser Results by Salesperson
UNITS SOLD
Andy
Chloe
Daniel
Diana
Sophia
Column Chart
Pie Chart

#1. Named argument

PHP 7

PHP 7 supports the name argument that means it passes values with parameter name.

PHP 8



In PHP 8 it specifies the required parameters and the argument is independent.

#2. Attributes

PHP 7

PHP 7 has the functionality to add metadata in code.

PHP 8

In PHP 8 instead of annotations, we can use structured metadata.

#2. Attributes

PHP 7

PHP 7 has the functionality to add metadata in code.

PHP 8

In PHP 8 instead of annotations, we can use structured metadata.

#3. Promotion of constructor

PHP 7

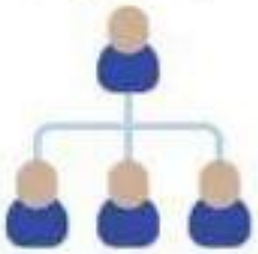
Basically, it is a simple class and is not used in PHP 7.

PHP 8

PHP 8 supports the construction and is used to describe the data structure.

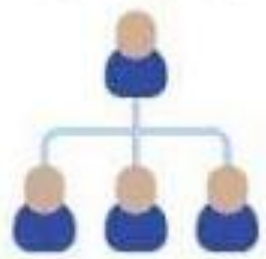
#4. Union type

PHP 7



In PHP 7 we need to use annotations.

PHP 8



But in PHP 7 we can use native type and that can be validated at runtime.

#5. Match expression

PHP 7

PHP 7 supports the match expression and that can be stored in variable.

PHP 8

PHP 8 supports the match expression and that can be stored in a variable.

#6. Null

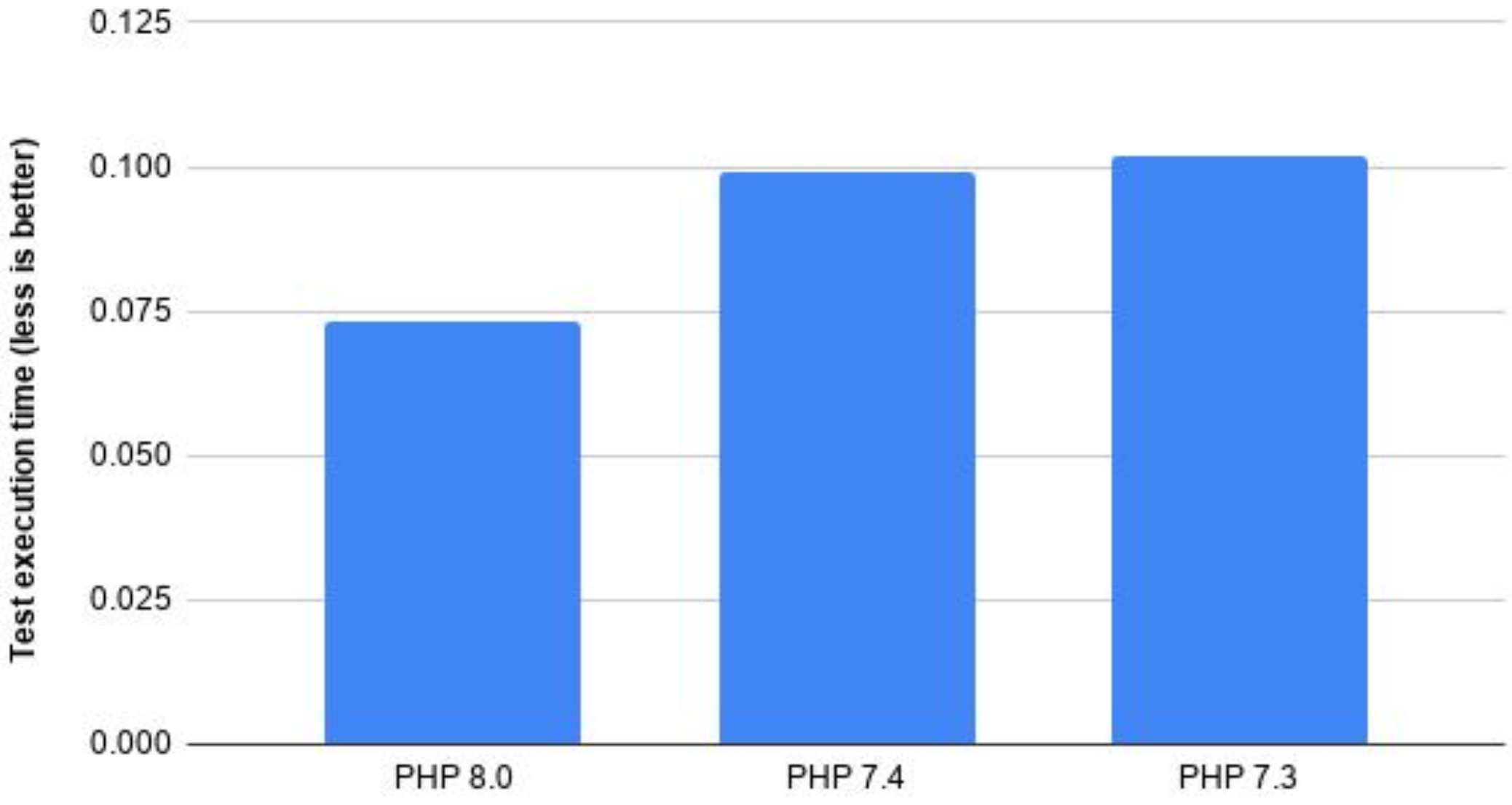
PHP 7

In PHP 7 we need to check null conditions.

PHP 8

In PHP 7 we can use chain with nullsafe operator.

www.educba.com


PHP 8.0 performance vs PHP 7.4 vs PHP 7.3
Test execution time (less is better)
0.125
0.100
0.075
0.050
0.025
0.000
PHP 8.0
PHP 7.4
PHP 7.3

Domande?