

Programmazione a oggetti

Corso Backend System Integrator
Modulo **Programmazione PHP**

Docente: Dott. Enrico Zimuel

in collaborazione con:



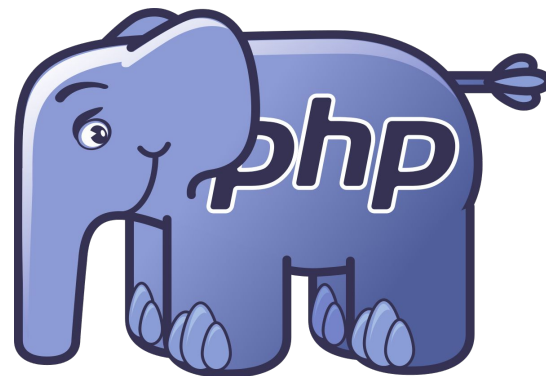
per una crescita intelligente,
sostenibile ed inclusiva

www.regione.piemonte.it/europa2020

INIZIATIVA CO-FINANZIATA CON FSE

Programma

- Introduzione alla OOP
- Le classi in PHP
 - Proprietà
 - Metodi
 - Costanti
- Costruttore di classe
- Dipendenze implicite/esplicite
- Copia di un oggetto



OOP

- La programmazione a oggetti (OOP, Object Oriented Programming) è una metodologia di sviluppo software molto utilizzata negli ultimi anni perché consente di scrivere codice:
 - **leggibile**
 - **strutturato**
 - **di facile manutenzione**

Principi della OOP

- Due dei principi fondamentali della OOP sono:
 - **Astrazione**, ossia la capacità di creare del codice in grado di risolvere una classe di problemi
 - **Incapsulamento**, ossia la possibilità di isolare (rendere autonomo) il codice rispetto al resto del programma

Classe

- Una **classe** è un'astrazione di un'entità contenente le possibili **azioni** e le sue **proprietà**
- Ad esempio la classe delle automobili può essere definita tramite le azioni: accendere, spegnere, accelerare, frenare e le proprietà: cilindrata, colore, alimentazione, classe, etc

Azioni e proprietà

- Le **azioni** di una classe sono definite tramite **funzioni** (detti anche metodi)
- Le **proprietà** di una classe sono definite tramite delle **variabili**

Classi in PHP

- Una classe è definita con il costrutto **class**

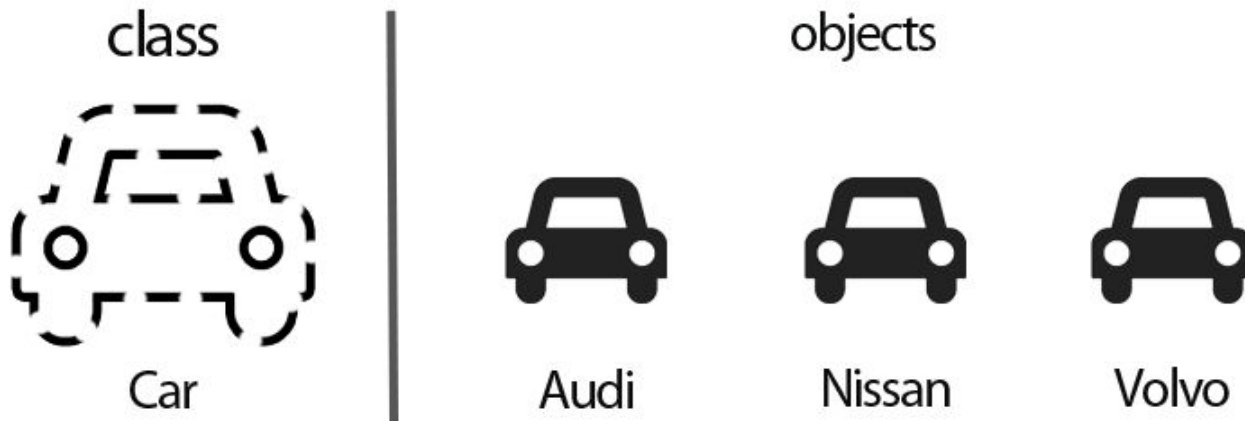
```
class Studente
{
    public string $nome;
    public string $cognome;

    public function hello(): string
    {
        return 'hello';
    }
}
```

Oggetti

- Una classe è soltanto una definizione di un insieme di **funzioni** (metodi) e **variabili** (proprietà)
- Per poter utilizzare una classe è necessario istanziarla, ossia creare un **oggetto** della classe
- In PHP si utilizza l'istruzione ***new* <classe>**, dove <classe> è il nome della classe

Classe vs. oggetti



Esempio

```
$alunno = new Studente;  
$alunno->nome = 'Mario';  
$alunno->cognome = 'Rossi';  
echo $alunno->hello();
```

L'operatore  viene utilizzato per accedere alle proprietà/metodi della classe

\$this

- **\$this** è una pseudo variabile in PHP che indica l'istanza della classe
- Viene utilizzata all'interno di una classe per accedere a proprietà o metodi della classe

Esempio

```
class Studente
{
    public string $nome;
    public string $cognome;

    public function getNomeCompleto(): string
    {
        return $this->nome . ' ' . $this->cognome;
    }
}
```

Nome di una classe

- Il nome di una classe PHP, così come il nome di una funzione, sono **case insensitive**
- Ad esempio, il codice seguente istanzierà 4 oggetti della stessa classe **Studente**:

```
$alunno1 = new Studente;  
$alunno2 = new STUDENTE;  
$alunno3 = new StUdEnTe;  
$alunno4 = new STUdente;
```

Proprietà di una classe

- Le proprietà di una classe sono le sue variabili
- Possono essere di 3 tipi:
 - **public**, visibili all'esterno della classe;
 - **private**, visibili solo all'interno della classe;
 - **protected**, visibili all'interno e a chi eredita la classe;

Esempio

```
class Punto {  
    public $x;  
    private $y;  
    protected $z;  
}  
  
$a = new Punto;  
$a->x = 1;  
$a->y = 2; // Fatal error: cannot access private property  
$a->z = 1; // Fatal error: cannot access protected property
```

Metodi di una classe

- Anche i metodi di una classe possono essere di 3 tipi:
 - **public**, eseguibili all'esterno della classe;
 - **private**, eseguibili solo all'interno;
 - **protected**, eseguibili all'interno e da chi eredita la classe;
- Il PHP utilizza il tipo **public** come valore predefinito, nel caso non venga specificato nessun tipo

Costruttore

- A volte, per poter creare un oggetto di una classe è necessario e effettuare un'operazione di **inizializzazione** (di costruzione)
- Questa operazione può essere specificata tramite il metodo **__construct**
- Il metodo **__construct()** viene invocato quando si crea un nuovo oggetto (con il costrutto **new**)

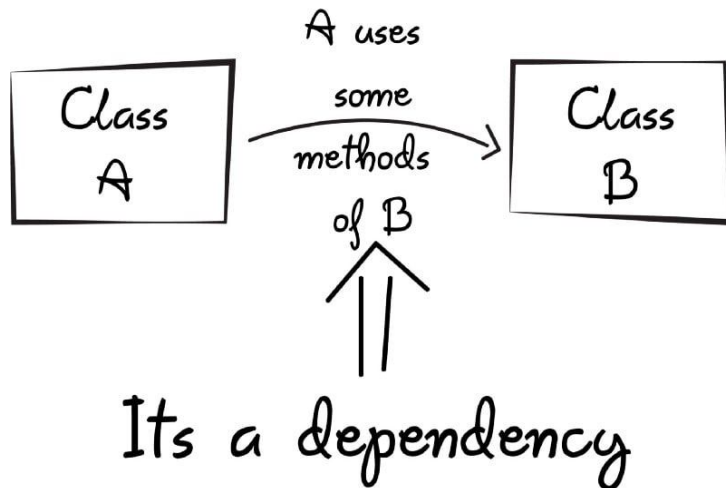
Esempio

```
class Studente
{
    private PDO $pdo; // PDO è l'oggetto per collegarsi al database
    public function __construct() {
        $this->pdo = new PDO(/* .. */);
    }
    public function getEmail(): string {
        $sth = $this->pdo->query('SELECT ...');
        return $sth->fetch();
    }
}

$alunno = new Studente; // viene eseguito __construct()
```

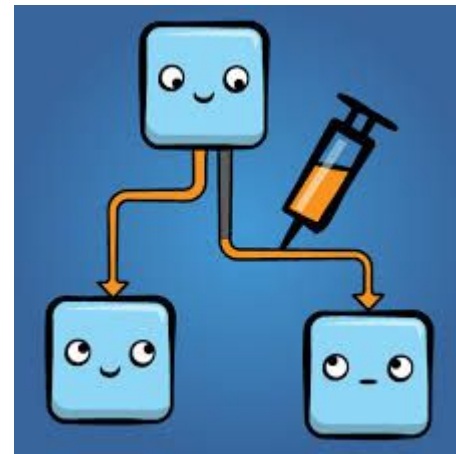
Dipendenza implicita

- La classe **Student** appena creata dipende dalla classe **PDO**
- C'è una dipendenza stretta tra queste due classi; la creazione della classe PDO è delegata alla classe Student, non è possibile variarla



Dipendenza esplicita

- Una buona pratica di programmazione è quella di passare le dipendenze di una classe in costruzione
- E' preferibile creare le dipendenze tra classi in maniera **esplicita** per facilitare l'organizzazione e l'interoperabilità del codice
- Questa pratica è chiamata [Dependency Injection](#)



Esempio

```
class Studente
{
    private PDO $pdo; // PDO è l'oggetto per collegarsi al database
    public function __construct(PDO $pdo) {
        $this->pdo = $pdo;
    }
    public function getEmail(): string {
        $sth = $this->pdo->query('SELECT ...');
        return $sth->fetch();
    }
}

$db = new PDO(/* ... */);
$alunno = new Studente($db); // viene eseguito __construct()
```

Esercizio

- Creare una classe che gestisca i dati di uno studente (nome, cognome, email, anno di nascita), i corsi seguiti con i relativi voti d'esame
- Creare un metodo **getMediaEsami(): ?float** che restituisca la media degli esami sostenuti

Costanti di classe

- E' possibile definire una costante in una classe tramite l'istruzione **const**
- E' possibile specificare anche il tipo di una costante di classe:
 - **public**, visibile all'esterno della classe;
 - **private**, visibile solo all'interno della classe;
 - **protected**, visibile all'interno e a chi eredita la classe;

Esempio

```
class Studente
{
    public const VOTO_MAX = 30;
    protected const VOTO_MIN = 0;
    private const MAX_NUM_ESAMI = 20;
}

echo Studente::VOTO_MAX;
echo Studente::VOTO_MIN; // Fatal error: Cannot access protected const
echo Studente::MAX_NUM_ESAMI; // Fatal error: Cannot access private const
```


Esercizio

Aggiungere alla classe `Studente` dell'esercizio precedente il controllo sull'inserimento di un voto valido tra 0 e 30 e di un numero massimo di corsi pari a 20, tramite delle costanti di classe (e.s. `MAX_VOTO`)

Copia di un oggetto

- Quando si prova ad assegnare un oggetto ad un'altra variabile si ottiene una copia per riferimento
- L'oggetto non viene di fatto copiato, è copiato solo il puntatore all'indirizzo di memoria
- Di fatto, le due variabili rappresentano lo stesso oggetto

Esempio

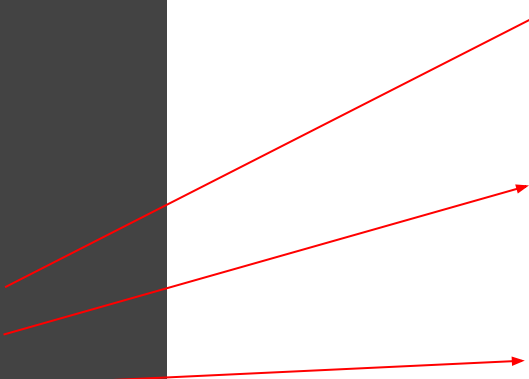
```
Class A {  
    public $i = 1;  
}  
$a = new A();  
$b = $a; // copia oggetto  
$b->i++;  
printf("a=%d, b=%d", $a->i, $b->i); // a=2, b=2
```

Clonare un oggetto

- Se desidero avere una copia di un oggetto in un'altra variabile devo clonare l'oggetto con l'utilizzo dell'operatore **clone**

```
Class A {
    public $i = 1;
}
$a = new A();
$b = $a;
$c = clone $a;
$b->i++;
var_dump($a);
var_dump($b);
var_dump($c);
```

```
object(A) #1 (1) {
    ["i"]=>
    int(2)
}
object(A) #1 (1) {
    ["i"]=>
    int(2)
}
object(A) #2 (1) {
    ["i"]=>
    int(1)
}
```



Esercizio (da consegnare)

- Creare una classe **ShoppingCart** che gestisca un carrello di un sito e-commerce, con i seguenti metodi:
 - `addProduct(string $product, int $quantity): bool`
 - `removeProduct(string $product): bool`
 - `changeQuantity(string $product, int $quantity): bool`
 - `removeAllProducts(): bool`

Grazie dell'attenzione!

Per informazioni:

enrico.zimuel@its-ictpiemonte.it