Ereditarietà OOP PHP

Slide tratte da:

W3CSCHOOLS.COM

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

Ereditarietà in PHP

- L'ereditarietà nella programmazione orientata agli oggetti (OOP) rappresenta la capacità di una classe di ereditare proprietà e metodi da un'altra classe genitore.
- In PHP, si può ottenere l'ereditarietà utilizzando la parola chiave "`extends`" per definire una classe figlia che deriva da una classe genitore.

Esempio (1) di ereditarietà

PHP

```php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

// Strawberry is inherited from Fruit
class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}
$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
```

Esempio di ereditarietà

- La classe `Strawberry` eredita da `Fruit`, ottenendo accesso alle sue proprietà pubbliche e metodi.
- La classe figlia può anche avere proprietà e metodi propri, come, p.e., il metodo `message()`.

- Il modificatore di accesso "protected" permette di accedere alle proprietà o ai metodi all'interno della classe e dalle classi derivate da quella classe.
- È possibile accedere alle proprietà e ai metodi "protected" all'interno della classe figlia.

Esempio (2) di ereditarietà

PHP

```
class Fruit {
  public $name;
  public $color;

  protected function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
    $this -> intro(); // OK. intro() is protected
  }
}

// Try to call all three methods from outside class
$strawberry = new Strawberry("Strawberry", "red");  // OK. __construct() is public
$strawberry->message();  // OK. message() is public
$strawberry->intro(); // ERROR. intro() is protected
```


Overriding dei metodi

- In OOP, è possibile sovrascrivere i metodi ereditati ridefinendoli con lo stesso nome nella classe figlia.
- Ciò consente di personalizzare il comportamento dei metodi nelle classi derivate.

Esempio di overriding

PHP

```
class Fruit {
  public $name;
  public $color;

  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

class Strawberry extends Fruit {
  public $weight;

  public function intro() {
    echo "The fruit is {$this->name}, the color is {$this->color}, and the weight is
{$this->weight} gram.";
  }
}

$strawberry = new Strawberry("Strawberry", "red", 50);
$strawberry->intro();
```

La parola chiave `final`

- La parola chiave `final` può essere utilizzata per impedire l'ereditarietà della classe o per impedire l'override dei metodi.

Esempio di classe "`final`"

PHP

```
final class Fruit {
  // some code
}

// will result in error
class Strawberry extends Fruit {
  // some code
}
```


Esempio di metodo "final"

PHP

```
class Fruit {
  final public function intro() {
    // some code
  }
}

class Strawberry extends Fruit {
  // will result in error
  public function intro() {
    // some code
  }
}
```


Costanti di classe

- Le costanti di classe sono utili per definire dati costanti all'interno di una classe.
- Una costante di classe viene dichiarata con la parola chiave "`const`" all'interno di una classe.

- È possibile accedere a una costante di classe all'esterno della classe utilizzando "`NOME_CLASSE::NOME_COSTANTE`".
- È possibile accedere a una costante di classe dall'interno della classe utilizzando "`self::NOME_COSTANTE`".

Esempio di costanti di classe

PHP

```
class Goodbye {
  const LEAVING_MESSAGE = "Goodbye!!";

  public function byebye() {
    echo self::LEAVING_MESSAGE;// accesso interno della classe
  }
}

echo Goodbye::LEAVING_MESSAGE; // accesso esterno della classe
```

Metodi statici

- I metodi statici possono essere chiamati direttamente, senza creare un'istanza della classe.
- Sono dichiarati con la parola chiave "`static`".
- Si richiamano all'esterno della classe con `NOME_CLASSE::NOME_METODO();`
- Si richiamano all'interno della classe con `SELF::NOME_METODO();`
- Se il metodo è pubblico può essere richiamato anche da un'altra classe con NOME_CLASSE::NOME_METODO();
- È possibile chiamare un metodo statico da una classe figlia utilizzando la parola chiave "`parent`".

Esempio di metodo statico

PHP

```
class greeting {
  public static function welcome() { // Definizione metodo statico
    echo "Hello World!";
  }

  public function __construct() {
    self::welcome(); // Chiamata all'interno della classe
  }
}
class altraclasse {
  public function message() {
    greeting::welcome(); // Chiamata all'interno di un'altra classe
  }
}

greeting::welcome(); // Chiamata all'esterno della classe
```


Proprietà statiche

- Le proprietà statiche possono essere chiamate direttamente, senza creare un'istanza della classe.
- Sono dichiarate con la parola chiave "`static`".
- Si richiamano all'esterno della classe con `NOME_CLASSE::NOME_PROPRIETA;`
- Si richiamano all'interno della classe con `SELF::NOME_PROPRIETA;`
- Se il metodo è pubblico può essere richiamato anche da un'altra classe con NOME_CLASSE::NOME_PROPRIETA;
- È possibile chiamare una proprietà statica da una classe figlia utilizzando la parola chiave "`parent`".

Esempio di metodo statico

PHP

```
class pi {
  public static $value=3.14159; // definisco proprietà statica
}

class x extends pi {
  public function xStatic() {
    return parent::$value; // accedo tramite classe figlia (all'interno)
  }
}

echo pi::$value; // accedo all'esterno della classe
echo x::$value; // accedo tramite classe figlia (all'esterno)

$x = new x();
echo $x->xStatic(); // accedo tramite metodo oggetto della classe figlia
```

Riepilogo

- L'ereditarietà permette di condividere proprietà e metodi tra classi genitore e classi figlie.
- I metodi statici possono essere chiamati direttamente senza creare istanze di classe.
- Le costanti di classe forniscono dati costanti all'interno di una classe.

Abbiamo finito!

Ora è importante fare parecchi esercizi!