Operatori e tipi di dato in PHP

In PHP, a partire dalla versione 7 e migliorato in PHP 8, è possibile indicare esplicitamente i tipi di dato per variabili, parametri, e valori di ritorno di una funzione. Questo meccanismo è conosciuto come **"Type Hinting"** e aiuta a garantire che il codice sia più prevedibile e meno incline a errori, poiché PHP verifica se i valori rispettano il tipo di dato specificato.

- **Tipi scalari**:
  - `int` (numeri interi)
  - `float` (numeri in virgola mobile)
  - `string` (stringhe)
  - `bool` (valori booleani)
- **Tipi composti**:
  - `array` (array)
  - `object` (oggetti di una classe specifica)
  - `callable` (funzioni o metodi passati come argomenti)
- **Tipi speciali**:
  - `mixed` (qualsiasi tipo di dato, introdotto in PHP 8)
  - `void` (nessun valore di ritorno)
  - `null` (assenza di valore)

Esempi di Utilizzo del Type Hinting

1. Indicare il Tipo per i Parametri di Funzione

È possibile indicare il tipo dei parametri di una funzione per garantire che i valori passati siano del tipo corretto.

Esempio:

```php
function sommaNumeri(int $a, int $b): int {
    return $a + $b;
}

echo sommaNumeri(3, 4); // Output: 7
```

In questo esempio, i parametri `$a` e `$b` devono essere di tipo `int`. Se si passa un valore di tipo diverso, PHP genererà un errore.

2. Indicare il Tipo di Ritorno della Funzione

PHP permette anche di specificare il tipo di dato che una funzione deve restituire.

Esempio:

```php
function getNomeCompleto(string $nome, string $cognome): string {
    return $nome . ' ' . $cognome;
}

echo getNomeCompleto('Mario', 'Rossi'); // Output: Mario Rossi
```

Qui, la funzione `getNomeCompleto` garantisce che il risultato sarà sempre una `string`.

3. Tipi Nullable

A volte potrebbe essere necessario che un parametro o il valore di ritorno possano essere `null`. In questo caso, si può usare il prefisso `?` prima del tipo di dato.

Esempio:

```
function ottieniEtà(?int $età): string {
    return $età ? "L'età è $età" : "Età non disponibile";
}

echo ottieniEtà(null); // Output: Età non disponibile
```

In questo caso, il parametro `$età` può essere sia un `int` che `null`.

4. Uso del Tipo `object`

Dal PHP 7.2 in poi, è possibile indicare un tipo generico `object` per accettare qualsiasi oggetto come parametro o valore di ritorno.

Esempio:

```php
function elaboraOggetto(object $obj): void {
    echo get_class($obj);
}

elaboraOggetto(new DateTime()); // Output: DateTime
```

Qui, la funzione accetta qualsiasi oggetto e restituisce il nome della sua classe.

5. Uso del Tipo `mixed`

Introdotto in PHP 8, il tipo `mixed` indica che un parametro o un valore di ritorno possono essere di qualsiasi tipo. È utile quando non si conosce in anticipo il tipo preciso.

Esempio:

```php
function accettaTutto(mixed $dato): mixed {
    return $dato;
}

echo accettaTutto(123); // Output: 123
echo accettaTutto("Ciao"); // Output: Ciao
```

6. Uso del Tipo `void`

Il tipo `void` indica che una funzione non deve restituire alcun valore.

Esempio:

```
function saluta(): void {
    echo "Ciao!";
}

saluta(); // Output: Ciao!
```

La funzione `saluta` non restituisce alcun valore.

Coercizione dei Tipi

Se il tipo di dato passato non corrisponde esattamente a quello specificato, PHP tenta di eseguire una **coercizione dei tipi** (trasformazione automatica) quando possibile. Tuttavia, a partire da PHP 7.0, è possibile forzare la verifica stretta del tipo usando la dichiarazione `declare(strict_types=1)` all'inizio del file PHP.

Esempio con coercizione dei tipi:

```php
function somma(int $a, int $b): int {
    return $a + $b;
}

echo somma(2.5, 3.7); // Output: 5 (i numeri in virgola mobile vengono convertiti in interi)
```

Esempio con tipi stretti:

```php
declare(strict_types=1);

function somma(int $a, int $b): int {
    return $a + $b;
}

echo somma(2.5, 3.7); // Errore, non sono interi
```

Vantaggi dell'Indicazione dei Tipi

1. **Miglior controllo**: Aiuta a prevenire errori in fase di esecuzione verificando che i valori passati e ritornati rispettino il tipo atteso.
2. **Codice più leggibile**: Specificare il tipo dei parametri e dei valori di ritorno rende il codice più facile da capire.
3. **Performance**: Sebbene la differenza possa non essere immediatamente evidente, l'uso del type hinting può migliorare le performance in alcuni contesti.

Conclusione

L'indicazione dei tipi di dato in PHP permette di scrivere codice più robusto e manutenibile, specialmente in progetti di grandi dimensioni o collaborativi. Sfruttando i miglioramenti di PHP 8, come il tipo `mixed`, il nullsafe operator (`?->`), e la gestione dei tipi nullable, il codice diventa più sicuro e facile da gestire.

Operatori

Operatori in PHP 8

Gli operatori sono simboli che eseguono operazioni su variabili o valori. PHP include diversi tipi di operatori: aritmetici, di confronto, logici, di assegnazione, ecc. In PHP 8 sono stati introdotti o migliorati alcuni operatori chiave che andremo a spiegare nel dettaglio.

Operatori Aritmetici

Gli operatori aritmetici eseguono operazioni matematiche sui numeri:

- `+` (addizione)
- `-` (sottrazione)
- `*` (moltiplicazione)
- `/` (divisione)
- `%` (modulo)

Operatori di Confronto

Questi operatori confrontano due valori:

- `==` (uguale a)
- `!=` o `<>` (diverso da)
- `===` (identico, sia per valore che per tipo)
- `!==` (non identico, differente per valore o tipo)
- `<` (minore di)
- `>` (maggiore di)
- `<=` (minore o uguale)
- `>=` (maggiore o uguale)

Operatore Spaziale (<=>)

Introdotto in PHP 7, l'operatore "spaziale" (o nave spaziale) `<=>` confronta due valori e restituisce:

- `0` se i due valori sono uguali,
- `-1` se il primo valore è minore,
- `1` se il primo valore è maggiore.

Esempio:

```
echo 1 <=> 2; // -1
echo 2 <=> 2; // 0
echo 3 <=> 2; // 1
```

Operatore di Fusione dei Valori Nulli (??)

L'operatore `??` (null coalescing) viene utilizzato per restituire il primo valore non nullo tra due o più valori. Questo operatore è utile per fornire un valore di fallback se una variabile non è stata impostata.

Esempio:

```
$valore = $variabileNonDefinita ?? 'valore di default';
```

Se `$variabileNonDefinita` è `null` o non esiste, verrà usato `'valore di default'`.

Operatore Nullsafe (?. o ?->)

Introdotto in PHP 8, l'operatore `?->` (nullsafe) permette di accedere alle proprietà o chiamare metodi di un oggetto in modo sicuro, evitando errori se l'oggetto è `null`. Questo operatore controlla se l'oggetto esiste prima di accedere alla proprietà o metodo. Se l'oggetto è `null`, il risultato sarà automaticamente `null`, evitando un errore di accesso.

Esempio:

```
$risultato = $oggetto?->metodo();
```

In questo caso, se `$oggetto` è `null`, non verrà sollevata un'eccezione, ma `$risultato` sarà semplicemente `null`.

Operatore Elvis (?:)

L'operatore Elvis è una forma abbreviata di `if`. Viene utilizzato per verificare se una variabile o un'espressione è vuota o `false`. Se sì, restituisce un valore alternativo.

Esempio:

```
$risultato = $valore ?: 'valore di default';
```

Se `$valore` è `false` o vuoto, allora verrà usato `'valore di default'`.

Operatori Logici

Gli operatori logici sono utilizzati per valutare condizioni booleane:

- `&&` (AND)
- `||` (OR)
- `!` (NOT)

Esempio:

```
if ($a && $b) {
    echo "Entrambe sono vere";
}
```

Operatori di Assegnazione

Gli operatori di assegnazione attribuiscono un valore a una variabile:

- `=` (assegnazione semplice)
- `+=`, `-=`, `*=`, `/=` (assegnazione combinata con operatori aritmetici)

Esempio:

```
$x = 10;
$x += 5; // Equivalente a $x = $x + 5;
```

Nuovi operatori in php 8

In PHP 8, come nelle versioni precedenti, gli **operatori** sono utilizzati per eseguire operazioni su variabili e valori. Oltre agli operatori tradizionali, PHP ha introdotto nuovi operatori che semplificano il codice e migliorano la leggibilità. Di seguito, una spiegazione dei principali operatori, inclusi i nuovi introdotti in PHP 7 e PHP 8:

1. Operatore di confronto "spaceship" (`<=>`)

L'operatore "spaceship" (`<=>`) è un operatore di confronto introdotto in PHP 7, che ritorna:

- `-1` se il primo operando è minore del secondo,
- `0` se sono uguali,
- `1` se il primo è maggiore del secondo.

Questo operatore è molto utile nelle operazioni di ordinamento.

Esempio:

```
echo 3 <=> 4;  // Output: -1 (3 è minore di 4)
echo 3 <=> 3;  // Output: 0 (3 è uguale a 3)
echo 4 <=> 3;  // Output: 1 (4 è maggiore di 3)
```

2. Operatore null coalescing (`??`)

L'operatore null coalescing (`??`) è stato introdotto in PHP 7 e permette di controllare se una variabile esiste ed è diversa da `null`. Se la variabile è definita e non è `null`, viene restituito il suo valore; altrimenti, viene restituito un valore alternativo.

Esempio:

```
$val = $_GET['name'] ?? 'Ospite';
// Se 'name' non è definito in $_GET, il valore di $val sarà 'Ospite'
```

3. Operatore nullsafe (`?->`) (PHP 8)

L'operatore nullsafe (`?->`) è una delle novità di PHP 8. Questo operatore evita di lanciare errori quando si tenta di accedere a proprietà o metodi su un oggetto che potrebbe essere `null`. Se l'oggetto è `null`, l'intera espressione restituisce `null`, anziché generare un errore fatale.

Esempio:

```
$utente = null;
echo $utente?->getNome();  // Non genera errore, semplicemente restituisce null
```

4. Operatore ternario (`?:`)

L'operatore ternario (`?:`) è una forma abbreviata dell'`if`. È composto da tre parti: la condizione, il risultato se la condizione è vera, e il risultato se la condizione è falsa.

Esempio:

```
$età = 20;
$messaggio = ($età >= 18) ? 'Sei maggiorenne' : 'Sei minorenne';
// Se $età è maggiore o uguale a 18, $messaggio sarà "Sei maggiorenne"
```

In alternativa, se non si vuole specificare il secondo operando (cioè il valore da restituire se la condizione è vera), si può usare una versione abbreviata:

```
$nome = $utente['nome'] ?: 'Ospite';
// Se $utente['nome'] è null o vuoto, viene assegnato 'Ospite'
```

5. Operatore di fusione nulla abbreviato (`??=`)

Questo operatore combina la logica del null coalescing con l'assegnazione. Se una variabile non è impostata o è `null`, viene assegnato un valore predefinito.

Esempio:

```
$nome ??= 'Ospite';
// Se $nome non è definito o è null, viene impostato su 'Ospite'
```

6. Operatore di concatenazione delle stringhe (`.`)

Questo operatore permette di concatenare stringhe in PHP.

Esempio:

```
$saluto = "Ciao, " . $nome . "!";
```

7. Operatore di assegnazione di concatenazione (`.=`)

L'operatore `.=`, invece, aggiunge una stringa a una variabile esistente.

Esempio:

```
$saluto = "Ciao, ";
$saluto .= "Mondo!";  // Risultato: "Ciao, Mondo!"
```