Introduzione a PHP

Programmazione

PHP (acronimo ricorsivo di PHP: Hypertext Preprocessor) è un linguaggio di scripting lato server progettato principalmente per lo sviluppo di applicazioni web dinamiche.

Viene eseguito sul server e consente di generare contenuti HTML in modo dinamico, gestire dati provenienti da form, interagire con database (come MySQL o PostgreSQL), gestire sessioni e autenticazioni.

Grazie alla sua sintassi semplice, alla vasta diffusione nei contesti web e all'ampio ecosistema di framework e CMS (come WordPress), PHP è uno dei linguaggi più utilizzati per la realizzazione di siti e applicazioni web backend.

- **Lato server (Server-side scripting)**

PHP viene eseguito sul server e non nel browser dell'utente. Questo significa che il codice PHP non è visibile al client e viene usato per generare dinamicamente l'HTML finale, garantendo maggiore sicurezza e controllo della logica applicativa.

- **Integrazione nativa con HTML**

PHP è progettato per essere facilmente integrato all'interno delle pagine HTML. Questo permette di mescolare markup e logica applicativa in modo diretto, rendendo semplice la creazione di pagine web dinamiche basate su dati e condizioni.

- **Interazione con database**

Una delle forze principali di PHP è la sua capacità di comunicare in modo efficiente con i database relazionali (come MySQL, MariaDB, PostgreSQL). Consente operazioni CRUD complete ed è alla base di applicazioni data-driven come CMS, e-commerce e sistemi gestionali.

- **Ampio ecosistema e portabilità**

PHP è open source, multipiattaforma e supportato da un vastissimo ecosistema di librerie, framework e CMS. Può essere eseguito su diversi sistemi operativi e server web, rendendolo una scelta flessibile e ampiamente adottata nello sviluppo web backend.

Ecco i 6 elementi fondamentali di PHP, come il simbolo $, spiegati in modo semplice e didattico:

- **<?php ... ?> – Tag PHP**

Indicano l'inizio e la fine di uno script PHP. Tutto il codice compreso tra questi tag viene interpretato dal server come PHP; il resto viene trattato come HTML.

- **$ – Variabili**

In PHP tutte le variabili iniziano con il simbolo $. Serve a indicare al linguaggio che si sta lavorando con una variabile e non con un valore letterale o una parola chiave.

- **; – Terminatore di istruzione**

Ogni istruzione PHP deve terminare con il punto e virgola. Serve al parser per capire dove finisce un comando e inizia il successivo.

- **// e /* */ – Commenti**

I commenti permettono di documentare il codice. // è usato per commenti su una sola riga, mentre /* */ per commenti su più righe. Non vengono eseguiti dal server.

- **echo – Output**

È il costrutto usato per stampare contenuti (testo, variabili, HTML) verso il browser. È uno degli strumenti più usati per mostrare risultati all'utente.

- **{ } – Blocchi di codice**

Le parentesi graffe delimitano blocchi di codice, ad esempio all'interno di if, else, for, while e function. Servono a raggruppare istruzioni che devono essere eseguite insieme.

Ecco i principali comandi (costrutti e funzioni base) di PHP, spiegati in modo essenziale e tecnico:

- **echo / print**

Servono per stampare output (testo, variabili, HTML) verso il browser. echo è leggermente più veloce e può stampare più argomenti, print restituisce un valore ed è meno usato.

- **if / else / elseif**

Gestiscono la logica condizionale del programma. Permettono di eseguire porzioni di codice solo se una condizione è vera, fondamentali per controllare flussi decisionali lato server.

- **for / while / foreach**

Sono i cicli di iterazione.

  - **for e while vengono usati per cicli numerici o condizionati**
  - **foreach è specifico per scorrere array e collezioni, molto comune in PHP**

- **function**

Permette di definire funzioni personalizzate, rendendo il codice riutilizzabile, modulare e più leggibile. Le funzioni possono accettare parametri e restituire valori.

- include / require

Consentono di includere file PHP esterni.

  - include genera solo un warning se il file manca
  - require blocca l’esecuzione in caso di errore

Sono alla base della strutturazione modulare delle applicazioni PHP.

- $_GET / $_POST

Sono superglobali usate per recuperare dati inviati dal client tramite form o URL. Fondamentali per la gestione dell’input utente nelle applicazioni web.

- **isset() / empty()**

Funzioni di controllo delle variabili.

- **isset() verifica se una variabile esiste ed è valorizzata**
- **empty() controlla se una variabile è vuota**

Utilissime per prevenire errori e validare input.

Per configurare PHP in modo semplice è consigliabile utilizzare un ambiente di sviluppo già pronto come XAMPP, WAMP o MAMP, che includono PHP, un server web (Apache) e un database.

Dopo aver installato il pacchetto, è sufficiente avviare il server web dal pannello di controllo e posizionare i file PHP nella cartella dedicata (ad esempio htdocs).

La configurazione principale di PHP avviene tramite il file php.ini, dove è possibile impostare parametri come la gestione degli errori, il fuso orario o le estensioni abilitate; una volta salvate le modifiche, basta riavviare il server per renderle effettive.

A questo punto PHP è pronto per essere utilizzato per sviluppare ed eseguire applicazioni web dinamiche in locale.

Il codice PHP è racchiuso tra i tag <?php e ?>, che indicano al server dove inizia e finisce lo script.

L’istruzione echo serve per inviare output al browser: in questo caso stampa la stringa di testo Hello World! quando la pagina viene eseguita dal server.

```
1. <?php
2. // Stampa un messaggio nel browser
3. echo "Hello World!";
4. ?>
```

In questo esempio viene dichiarata una variabile $eta.

Con la struttura if / else PHP verifica una condizione logica e, in base al risultato, utilizza echo per stampare un messaggio diverso nel browser.

È un tipico controllo lato server usato per validare dati o gestire permessi.

```
<?php
$eta = 20;

// Controllo con if
if ($eta >= 18) {
    echo "Sei maggiorenne";
} else {
    echo "Sei minorenne";
}
?>
```

Qui viene creato un array di nomi e una funzione personalizzata.

**Il ciclo foreach serve per scorrere tutti gli elementi dell'array, mentre echo stampa ogni valore andando a capo con
.**

La funzione incapsula la logica, rendendo il codice più ordinato e riutilizzabile, pratica fondamentale nello sviluppo in PHP.

```php
<?php
$nomi = ["Anna", "Luca", "Marco"];

// Definizione di una funzione
function stampaNomi($lista) {
    foreach ($lista as $nome) {
        echo $nome . "<br>";
    }
}

// Chiamata della funzione
stampaNomi($nomi);
?>
```

- **Dimenticare il simbolo $ nelle variabili**

In PHP tutte le variabili devono iniziare con $. Scrivere nome = "Mario"; invece di $nome = "Mario"; genera errori di parsing o comportamenti inattesi.

- **Confondere = con == o ===**

= è l'operatore di assegnazione, mentre == e === servono per il confronto. Usare = dentro una condizione if è un errore frequente che porta a risultati logici errati

- **Non controllare l’esistenza delle variabili ($_GET, $_POST)**

Accedere direttamente a $_GET o $_POST senza verificare con isset() o empty() può causare warning o errori. È fondamentale validare sempre l’input dell’utente.

- **Dimenticare il punto e virgola ;**

Ogni istruzione PHP deve terminare con ;. La sua assenza è una delle cause più comuni di errori di sintassi, spesso segnalata dal parser sulla riga successiva, rendendo il debug più difficile.

Il codice inizia con i tag PHP <?php ?>, che indicano al server di interpretare il contenuto come script PHP. Viene dichiarata una variabile $eta usando il simbolo $ e ogni istruzione termina con il punto e virgola ;

La funzione isset() viene usata per verificare l'esistenza della variabile, evitando errori. Con la struttura if / else viene gestita una logica condizionale che stampa un messaggio tramite echo.

Successivamente viene creato un array, che viene elaborato con un ciclo foreach all'interno di una funzione, dimostrando l'uso dei blocchi di codice { }. I commenti (//) rendono il codice più leggibile ma non vengono eseguiti.

Questo esempio rappresenta una base reale e completa del funzionamento di PHP lato server.

```php
<?php
// Dichiarazione di una variabile
$eta = 22;

// Controllo se la variabile esiste ed è valorizzata
if (isset($eta)) {

    // Condizione logica
    if ($eta >= 18) {
        echo "Accesso consentito<br>";
    } else {
        echo "Accesso negato<br>";
    }

    // Array di nomi
    $nomi = ["Anna", "Luca", "Marco"];

    // Funzione personalizzata
    function stampaNomi($lista) {
        foreach ($lista as $nome) {
            echo $nome . "<br>";
        }
    }

    // Chiamata della funzione
    stampaNomi($nomi);
}
?>
```