LINGUAGGIO PHP

linguaggio php

I comandi e le istruzioni php vengono scritti all'interno di pagine web, cioè in file di testo con particolari demarcatori.

```
<? php
//elenco istruzioni php
?>
```

Inizio e fine del blocco di codice.

```
<? php
/*
 * commento
/*
?>
```

Commento su più righe.
// serve per commentare su una sola riga.

Il file di testo deve essere salvato nel formato .**php**, includiamo quindi degli script nel php.
Il server si comporta nel seguente modo:
1. Legge il file di testo riga per riga
2. Se trova dei marcatori html li spedisce al browser
3. Se trova dei blocchi di codice php ne esegue l'interpretazione e restituisce i risultati al browser

Alcuni web server possono essere case sensitive per il nome della pagina.
Ogni istruzione può occupare una o più righe e termina sempre con il punto e virgola **;**

```
<? php
echo "stringa" ;
?>
```

Serve per inviare una stringa al browser.
Si possono aggiungere anche marcatori html.

```
<? php
print (stringa) ;
?>
```

Funzione di output

Differenza:
echo è in grado di gestire più stringhe alla volta, mentre print ne gestisce una sola.

Variabili

Le variabili sono identificate da un nome preceduto dal simbolo del dollaro **$**.
Sono case sensitive e possono contenere lettere, cifre e il carattere di sottolineatura.

I principali tipi sono:
1. Numero intero
2. Numero a virgola mobile
3. Stringa
4. Valori logici
5. Array
6. Oggetto

```
<? php
$nome = "Alessia";
$saluto = "buongiorno $nome" ;
?>
```

Metodo di assegnazione di valore a variabili che contengono una stringa racchiudendo il testo tra doppi apici "".
In questo modo le variabili presenti all'interno della stringa vengono espanse del loro nome.
Con gli apici singoli questo non avviene ''.

Le **sequenze di escape** seguenti devono essere racchiuse nei doppi apici:
1. \n ritorno a capo
2. \t tabulazione
3. \\ backslash
4. \$ simbolo del dollaro
5. \" doppi apici
6. \' apice singolo

Le stringhe possono essere concatenate utilizzando il punto **.** .

Operatori aritmetici: + , - , * , / , % , ++ , - - .

Operatori di assegnamento: = , += , -= , *= , %= , .= . Con i combinati posso usare il valore di una variabile all' interno di un'espressione senza indicarla esplicitamente. Es: $a += $b corrisponde $a = $a + $b.

Operatori di confronto: == , != , < , > , <= , >= .

Operatori logici: ! , && , | | , xor .

Il linguaggio php esegue automaticamente numerose conversioni di tipo.

```
<? php
strlen ($stringa);
?>
```

Riporta la lunghezza della stringa.

Array

Un array è rappresentato da una variabile che contiene un insieme di valori identificati da un indice.
L'**indice** con cui si accede all'array può essere un numero oppure una stringa.
Nello stesso array posso memorizzare valori di diverso tipo.

Negli **array numerici** l'indice è un numero intero che parte da 0.

```
<? php
$prezzi [0] = 4;
$prezzi [1] = 7;
?>
```

Una notazione equivalente per l'attribuzione dei dati ad un array è:
$prezzi = array (4 , 7) ;

Il valore speciale **null** indica che a quel dato non è stato assegnato alcun valore.

```
<? php
$persona ["nome"] = "Alessia" ;
$persona ["professione"] = "studente" ;
?>
```

Negli **array associativi** l'indice è la stringa che è indicata nei doppi apici.

Funzioni e variabili predefinite

Le *funzioni predefinite* sono:

1. **count($ar) :** restituisce il numero degli elementi di un array.
2. **rand(0, 1) :** restituisce un numero casuale tra quelli scritti.
3. **var_dump($ar[1]) :** restituisce tipo e valore di una variabile.
4. **isset($variabile) :** controlla se ad una variabile è stato assegnato un valore oppure no. Restituisce true in caso sia stato assegnato un valore oppure false nel caso non sia stato assegnato alcun valore oppure ha valore null.

Le *variabili predefinite* sono create direttamente dall'interprete php e sono raggruppate nei seguenti array associativi:

1. **$_GET:** contiene tutte le variabili passate allo script tramite la modalità GET.
2. **$_POST:** contiene tutte le variabili passate allo script tramite la modalità POST.
3. **$_SERVER:** contiene tutte le variabili passate allo script dal server Web.
4. **$_COOKIE:** contiene le variabili passate allo script tramite i cookie.
5. **$_SESSION:** contiene variabili utilizzate per implementare il concetto di sessione.

Strutture di controllo

```
<? php
if (condizione) {
    //istruzioni eseguite se la condizione è vera
} else {
    //istruzioni eseguite se la condizione è falsa
}
?>
```

Il ramo else della struttura di selezione può non essere presente.

```
<? php
while (condizione)
    {
    //istruzioni eseguite mentre la condizione si mantiene vera
    }
?>
```

Il controllo sulla condizione viene eseguito prima di ogni esecuzione del ciclo.

```
<? php
for (inizializzazione : condizione : aggiornamento) {
    //istruzioni eseguite se la condizione è vera
}
?>
```

Inizializzazione: istruzione da eseguire una sola volta prima dell'inizio del ciclo
Condizione: stabilisce se continuare o interrompere il ciclo.
Aggiornamento: istruzioni eseguite al termine di ogni iterazione.

Form

L'interazione con l'utente può essere gestita tramite **moduli del linguaggio html** che permettono la costruzione di una interfaccia grafica.

```
<form action = "nomedelfilephp.php" method = "get" oppure "post">
```

L'attributo **method** ha il compito di indicare al browser quale modalità deve usare per inviare i campi del modulo:

1. **get:** nell'url si può visualizzare ciò che è stato digitato.
2. **post:** può spedire grandi quantità di dati senza che questi vengano visualizzati dall'utente.

```
<form action = "nomedelfilephp.php" method = "get" oppure "post">
Dati : <input type = "text"  name = "dato"/>
<input type = "submit "  value = "cerca"/>
</form>
```

Per ogni campo crea un parametro avente come nome il nome indicato nell'attributo **name** del tag input e come valore ciò che l'utente ha inserito.
Usando si clicca sul pulsante cerca **submit** il browser richiama lo script .php indicato nell'intestazione della form action. Il browser aggiunge automaticamente alla richiesta tutti i campi present nel modulo.

$_SERVER['REQUEST_METHOD'] contiene il tipo di method usato.

Mysql

PhpMyAdmin è il programma più diffuso per l'utilizzo di mysql.
MySQL è un sistema per la gestione di basi di dati relazionali.

```
$mysqli = mysqli_connect("localhost", "sm", "","sm");
```

Connessione al server.

```
if (!$mysqli) { //
                    echo "Error: Unable to connect to MySQL.<br>";
                    echo "Debugging errno: " . mysqli_connect_errno() . "<br>";
                    echo "Debugging error: " . mysqli_connect_error() . "<br>";
                    exit;
```

Gestione dell'errore
errno: valore numerico del messaggio di errore.
error: testo del messaggio di errore sull'operazione.

```
$res = mysqli_query($mysqli, $sql);
```

Per eseguire la query.
mysqli contiene la connessione.
sql contiene il testo della query.

Le *funzioni* principali:

1. **mysqli_affected_rows ($mysqli):** restituisce il numero delle righe coinvolte nell'ultima operazione eseguita sulla connessione.
2. **mysqli_fetch_array($sql):** riceve come parametro la variabile contenente la query. Il valore di ritorno è un array in cui ogni elemento corrisponde al campo del record. Quando non ci sono più righe da esaminare restituisce il valore false. Viene fatto scorrere attraverso un ciclo while.
3. **mysqli_fetch_assoc:** caricamento di una riga della tabella in un array associativo.
4. **mysqli_fetch_row:** caricamento di una riga della tabella risultante come un array.

```
mysqli_close($mysqli);
```

Chiusura della connessione.