Guida base al linguaggio PHP

Cos'è PHP

PHP (acronimo di Hypertext Preprocessor) è uno dei più utilizzati linguaggi di programmazione server-side per realizzare pagine Web.
E' server-side perchè è interpretato dentro il server e non nel browser come accade per linguaggi di programmazione come javascript (che comunque ultimamente trova applicazione anche in ambiente server).

Creare un programma PHP

Un programma PHP può essere scritto tramite un comune editor di testo come blocco note, notepad++ ecc. Il codice PHP dovrà essere compreso tra “<?php” e “?>”. Il file andrà salvato con estensione .php. All'interno di un file PHP sarà possibile inserire anche codice HTML ma prima dell'apertura del PHP, quindi prima di “<?php” o dopo la sua chiusura, quindi dopo “?>” oppure utilizzando i tag HTML come stringhe da visualizzare tramite istruzione echo.

I tipi di dato

Php è un linguaggio a tipizzazione debole, ovvero non è necessario dichiarare il tipo di dato che la variabile conterrà, e una stessa variabile può contenere tipi di dato diversi: ad esempio prima può contenere un intero, poi un numero decimale o una stringa.
I principali tipi di dato semplice sono: integer (numero intero), float (numero frazionario), string (testo), boolean (valore booleano, ovvero che può assumere solo il valore vero o il valore falso).
String, in italiano stringa, è un tipo di dato che contiene tutto ciò che è una combinazioni di caratteri e numeri. Sostanzialmente quindi, stringa è tutto ciò che non può essere inquadrato negli altri tipi di dato semplice previsti in PHP.
Esistono anche tipi di dato complessi come Array e oggetti che verranno trattati più avanti

Le variabili

Una variabile è un “cassetto” di memoria nel quale viene memorizzato un valore che dovrà essere riutilizzato più avanti nel codice. Le variabili PHP iniziano con il carattere $ seguito dal nome scelto per la variabile (il nome della variabile può essere una combinazione di lettere, numeri e alcuni, ma non tutti, simboli; il nome deve iniziare sempre con una lettera).
Una variabile può contenere numeri (sia interi che frazionari), testo o strutture dati di tipo complesso.
Per assegnare un valore a una variabile si usa il simbolo =. Ad esempio per assegnare il valore 10 alla variabile $a si userà la seguente sintassi:
$a=10;
Per assegnare un valore di tipo stringa ad una variabile, il valore va posto all'interno di una coppia di apici singoli o doppi, ad esempio:

$a=”Giovanni”

L'istruzione PHP

Per istruzione php si intende un'operazione minima, spesso non divisibile in sottoparti, che attraverso il linguaggio di programmazione, viene richiesta al computer.
Un esempio di operazioni può essere il calcolo di una somma e il salvataggio del risultato in una variabile. L'**inizializzazione** di una variabile è l'operazione che consente di dare un valore iniziale a una variabile. Ogni istruzione PHP termina con il punto e virgola.

Le operazioni matematiche

Come in tutti i linguaggi di programmazione, anche in PHP è possibile effettuare operazioni matematiche. Per effettuare la somma si utilizzerà il carattere +, per la sottrazione -, per la divisione /, per la moltiplicazione *, per il resto %. Molto spesso per incrementare di uno il valore di una variabile si usa la sintassi $i++ che è da intendersi equivalente a $i=$i+1;

Operatori di confronto

Gli operatori di confronto consentono di definire dei test che verranno utilizzati da altri costrutti.
Gli operatoti di confronto sono: > (maggiore), < (minore), == (uguale), != (diverso), >= (maggiore o uguale), <= (minore o uguale).
Un test ha come possibili risultati soltanto due valori: vero o falso.

If

Il costrutto if consente di eseguire un blocco di codice o un altro a seconda del risultato di in test.
Un esempio di possibile uso è il seguente:

```
if($a>0)
{
	// blocco 1
}
else
{
	// blocco 2
}
```

Il codice nel blocco 1 viene eseguito se il test ha come esito vero mentre il codice nel blocco 2 viene eseguito se il test ha come esito falso. La sezione else con il relativo blocco di istruzioni può essere omessa se non è utile eseguire del codice in casi di test con esito falso.

I clicli

I clicli in informatica consentono di ripetere un blocco di operazioni diverse volte, finchè una condizione continua ad essere valida. Con il termine **iterazione** si intende la i-esmima ripetizione nella quale il ciclo si trova. I cicli più usati sono il ciclo for e il ciclo while.

Ciclo for

Un esempio di ciclo for è il seguente:

```
for ($i=0;$i<10;$i++)
{
        echo $i;
}
```

All'interno delle parentesi tonde vengono specificate le tre istruzioni che sono necessarie per configurare correttamente il ciclo for: la prima definisce e inizializza la variabile che verrà utilizzata dal ciclo, la seconda definisce la condizione di arresto, la terza l'incremento. Le tre istruzioni vanno interpretate in questo modo: il ciclo inizia con i uguale a 0, ad ogni iterazione i viene incrementata di uno, il ciclo si ferma quando i raggiunge il valore 10. Il blocco di istruzioni che viene eseguito ad ogni iterazione è quello racchiuso all'interno delle parentesi graffe. In questo caso il blocco è costituito da una sola istruzione: echo $i; **echo** è l'istruzione che consente di visualizzare, in questo caso il contenuto della variabile $i, sul monitor.

Ciclo While

Il ciclo while consente di ripetere un blocco di istruzioni finchè una condizione è valida.
Di seguito un esempio di una parte di codice PHP che contiene un ciclo while:

```
$a=0;
while($a<10)
{
        echo $a;
        $a++;
}
```

Il ciclo proposto visualizza il contenuto della variabile $a ripetendo tale azione tante volte quante la condizione posta dentro le parentesi tonde è verificata, ovvero fino a quando vale che $a<10;

Le funzioni

Le funzioni sono uno strumento dei linguaggi di programmazione per riutilizzare il codice. Una funzione consente di definire delle operazioni che devono essere effettuate su dei dati di ingresso (parametri di ingresso) le quali determineranno un risultato che verrà restituito dalla funzione. Una volta definita, la funzione potrà essere richiamata più volte nel programma consentendo di riutilizzare le operazioni che sono state definite al suo interno senza necessità che vengano riscritte più e più volte.
Una funzione è definita attraverso l'uso della parola “function” seguita da uno spazio e dal nome che si vuole dare alla funzione. Di seguito un esempio di funzione:

```
function somma($a,$b)
{
        $c=$a+$b;
        return $c;
}
```

In questo caso, la funzione ha come nome somma, prevede due parametri di ingresso ($a e $b) e restituisce il valore della variabile $c (che conterrà la somma dei valori delle variabili $a e $b). Quando una funzione dovrà essere utilizzata all'interno di una porzione di codice, dovrà essere

invocata specificandone anche i parametri di ingresso previsti dalla funzione, esempio:

```
$primo=1;
$secondo=8;
$risultato=somma($primo,$secondo);
echo $risultato;
```

Alcune funzioni sono predefinite in PHP e possono quindi essere utilizzate senza che vengano prima definite dal programmatore.

Tipi di dato complessi: gli array

Gli array consentono di inserire più valori all'interno della stessa variabile creando una sorta di elenco ordinato in cui ogni valore ha una posizione ben precisa individuata da un numero intero positivo. Il primo valore inserito nell'array occuperà la posizione 0. La posizione è chiamata indice dell'array. Affinchè una variabile possa contenere un array, deve essere inizializzata ad array nel modo seguente:
$a=array();
Per inserire un valore nell'array occorre esplicitare la posizione, quindi l'indice, nella quale si vuole inserire il valore. Con le istruzioni:
$a[0]=”Giovanni”;
$a[1]=”Pietro”;
verrà inserita la stringa “Giovanni” nella pozione 0 e la stringa “Pietro” nella posizione 1.
Per accedere al contenuto di ciascuna posizione di un array è possibile utilizzare un ciclo for per iterare tutte le posizioni dell'array. Se invece si vuole accedere a una specifica posizione dell'array, ad esempio si vuole visualizzare il contenuto dell'array nella posizione 1, si dovrà usare la seguente sintassi:
echo $a[1];
così come $b=$a[1]; copierà il valore della posizione 1 dell'array nella variabile $b (che non è un array).
La funzione count() restituisce il numero di elementi contenuti nell'array, quindi count($a), nel caso dell'esempio, restituirebbe 2.

Tipi di dato complessi: gli array associativi

Gli array associativi sono come i normali array con l'unica differenza che l'indice è di tipo stringa e non numerico. Di seguito un esempio di array associativo:
$a=array();
$a['nome']=”Giovanni”;
echo $a['nome'];

In questo esempio, la variabile $a viene inizializzata ad array, poi viene memorizzato il valore “Giovanni” nella posizione 'nome'. Nell'ultima riga, viene recuperato il valore contenuto nell'indice 'nome' dell'array e visualizzato tramite il comando echo.

Tipi di dato complessi: gli oggetti

L' oggetto è una struttura dati che è alla base del modo moderno di organizzare il codice.
Di fatti oggi si parla di programmazione ad oggetti proprio per indicare che l'architettura con la quale vengono disegnati i moderni programmi è basata su oggetti.
Un oggetto è un' istanza, quindi una concretizzazione di una classe. La **classe** è invece un oggetto astratto, un modello.
Una classe (rimarchiamo che la classe definisce l'oggetto) ha la particolare caratteristica di contenere al suo interno dati, chiamati **proprietà**, e funzioni che agiscono su questi dati, chiamate **metodi**.

La programmazione ad oggetti

Definizione di una classe

Possiamo quindi pensare ad una classe come a un contenitore di proprietà e metodi, della quale può essere creata un'istanza che servirà ad ospitare dati e i metodi che potranno essere usati su questi dati. Facciamo un esempio di definizione di una classe, che chiameremo conto, con la quale definiremo i dati e le operazioni elementari necessarie per gestire un conto corrente:

```
class conto {

private $giacenza;

public function __construct($importo)
{
	$this->$giacenza>=$importo
}

public function preleva($importo)
{
	if($this->$giacenza>=$importo)
	{
		$this->$giacenza=$this->$giacenza-$importo;
		return true;
	}
	else
	{
		return false;
	}
}

public function versa($importo)
{
	$this->$giacenza=$this->$giacenza+$importo;
	return $this->$giacenza;
}
```

}

La classe qui proposta ha una proprietà: $giacenza e tre metodi: __construct, preleva e versa. L'uso di $this-> all'interno dei metodi ha la finalità di puntare al valore della proprietà della classe.
Il metodo __construct è il metodo costruttore, il cui nome non è modificabile. La sua presenza non è obbligatoria, nel senso che è possibile definire classi che non abbiano il metodo costruttore.
Per essere utilizzata la classe deve essere istanziata, cioè il modello astratto che essa rappresenta deve essere utilizzato per riservare un'area di memoria del computer deputata ad ospitare i dati che la specifica istanza conterrà. Di seguito un esempio:

```
$conto_utente=new conto(1000);
$giacenza=$conto_utente->preleva(200);
```

La prima riga dell'esempio crea l'istanza della classe conto attraverso l'uso della parola “new”. All'invocazione di new, viene richiamato, se esiste, il metodo costruttore, che in questo caso setterà la giacenza al valore di 1000.
La seconda riga invoca il metodo preleva che agirà sull'attuale giacenza (1000) decurtando 200 e restituendo la nuova giacenza.
“private”, “public” sono i modificatori di accesso ovvero definiscono se la proprietà o il metodo può essere richiamato dall'esterno della classe o meno, in particolare usando public il metodo o la proprietà può essere richiamata dall'esterno, con private no. Esiste un terzo possibile valore di modificatore d'accesso, protected che però verrà trattato più avanti.
Se la proprietà $giacenza fosse stata dichiarata come public, dall'esterno della classe si sarebbe potuta usare la sintassi $conto_utente->giacenza per conoscere l'attuale giacenza.

Caratteristiche fondamentali della programmazione ad oggetti

La programmazione ad oggetti si caratterizza per:

- incapsulamento
- ereditarietà
- polimorfismo

L'incapsulamento è la capacità della classe di contenere al suo interno dati e metodi nascondendo all'esterno i dettagli implementativi.
L'ereditarietà è la possibilità che una classe venga estesa dichiarando una seconda classe che integra la prima e la estende mediante l'aggiunta di nuovi metodi e/o nuove proprietà.
Il polimorfismo consente nella possibilità di avere un metodo (ma anche più di uno) che ha lo stesso nome in più classi, che estendono la stessa classe base, però con differenti implementazioni, e quindi differenti comportamenti.

L'ereditarietà:

L'ereditarietà è una funzionalità basilare dei linguaggi di programmazione ad oggetti. Consiste nella possibilità di estendere una classe aggiungendo proprietà e metodi, consentendo quindi di creare delle classi più specializzate che estendono funzionalità già implementate nella classe

base.
Facciamo un esempio: se si vuole realizzare un software per la scuola che contenga informazioni su studenti e docenti, si potrebbe creare una classe base persona e due classi: docente e studente, che estendono la classe base persona. Nella classe base persona potremo definire la proprietà nome con i metodi per leggere e settare il nome, nella classe studente inseriremo solo la classe di appartenenza con il metodo per leggerla e settarla. Nella classe docente la materia insegnata con il metodo per leggerla e settarla. La classe studente e quella docente, estendendo la classe persona, erediteranno sia le proprietà che i metodi della classe base persona potendone quindi disporre come se fossero stati implementati all'interno di ciascuna di essa.
Definiamo la classe

```
class persona {
	private $nome;

	public function getNome()
	{
		return $this->nome;
	}

	public function setNome($nome)
	{
		$this->nome=$nome;
	}

}
```

ed ora la classe studente che estende la classe persona

```
class studente extends persona
{
	private $classe;

	public function getClasse()
	{
		return $this->classe;
	}

	public function setClasse($classe)
	{
		$this->classe=$classe;
	}

}
```

extends è la parola chiave che configura l'estensione della classe persona effettuata dalla classe studente.
Poichè studente estende persona, ne ingloba proprietà e metodi pertanto sarà possibile invocare

il metodo set Nome anche sulla classe studente.

Esempio:
$studente=new studente();
$studente->setNome(“Giovanni”);

echo $studente->getNome();