PHP

INTRODUZIONE

Una pagina Web statica, realizzata in HTML, è costituita da informazioni che non variano nel tempo e né tanto meno dalle azioni compite dall'utente, o almeno non variano finché il programmatore non le aggiorna manualmente.

Quando il client fa richiesta di una pagina Web statica, il server risponde fornendogli esattamente quanto richiesto senza fare elaborazioni.

In una pagina Web dinamica, invece, i contenuti cambiano in dipendenza di tanti fattori (tempo, utente, etc.). In questo caso il server utilizzerà delle tecnologie per elaborare la risposta da fornire al client.

Una pagina Web di questo tipo risulta essere più interattiva, e di fatti si parla di Web 2.0, in contrapposizione al Web 1.0 basato sulle pagine statiche. L'utente non è più un soggetto passivo a cui vengono soltanto inviate informazioni dal server, ma ha la possibilità, interagendo, di cambiare i contenuti della pagina stessa, oppure la pagina stessa si adatta alle caratteristiche, necessità, esigenze dell'utente. Casi tipici di siti Web dinamici sono i blog, i social network, gli Wiki, i motori di ricerca, le Home Banking, etc.

I linguaggi di programmazione dinamici (lato client e lato server), come JavaScript, ASP, PHP, etc., consentono di modificare automaticamente i contenuti di una pagina a seconda del tempo, dell'interazione dell'utente e del Web Master.

PREREQUISITI

Per realizzare siti Web o applicazioni Web in PHP occorre avere delle nozioni di base sui seguenti concetti:

- L'HTML, il linguaggio di base con cui sono scritte le pagine Web statiche.
- La Programmazione, in linguaggio C like.

I WEB SERVER

Il Web Server è un computer che ospita le pagine web che l'utente visita, qualsiasi sia il linguaggio di programmazione utilizzato per scriverle.

Su una tale macchina saranno quindi presenti dei file di vario genere ordinati ed archiviati in cartelle e sottocartelle. Alcuni file sono del tutto identici a quelli presenti sui PC utente (ad esempio file audio mp3, immagini in formato jpg, gif, etc.) altri, invece sono pagine HTML rese accessibili agli utenti attraverso il protocollo HTTP (HyperText Transfer Protocol) che è una modalità di trasferimento dei dati in una rete.

I computer che accedono ai file contenuti nel Web Server vengono detti Client.

L'accesso a tali file avviene attraverso l'URL (Universal Resource Locator) che indica la localizzazione di un file in rete e si compone nel seguente modo:

http://nomehost/percorso/nomefile

Quindi un URL si compone in questo modo: (http://) modalità di trasferimento dei dati, (nomehost) nome della macchina che contiene i file, (/percorso/) il percorso fra le cartelle, (nomefile) il nome del file (o risorsa) che si vuole visualizzare.

La visualizzazione di una pagina Web sul computer Client è resa possibile dall'elaborazione del codice scritto in HTML. È il browser del Client che si preoccupa di interpretare tale codice al fine di poter visualizzare correttamente i contenuti della pagina Web.

I linguaggi di programmazione detti "Lato Client" prevedono una elaborazione del loro codice attraverso il browser e questi sono fondamentalmente l'HTML, JavaScript e CSS. Infatti cliccando sul tasto destro del mouse su una qualsiasi pagina Web e selezionando HTML o codice sorgente (a seconda dei browser) sarà possibile vedere il codice elaborato dal browser. Quindi la rappresentazione a video di un sito Web, e alcune delle sue animazioni, sono generate dall'elaborazione che il browser eseguirà dei linguaggi lato client.

I linguaggi "Lato Server", come il PHP o ASP prevedono invece l'elaborazione diretta del Web Server il quale restituirà al Client, in genere al browser del Client, anziché il codice originale con cui è stato scritto il file (codice PHP per esempio), l'elaborazione del codice stesso. Il codice PHP, quindi, verrà interpretato dal Server e andrà a generare, a sua volta, un linguaggio lato Client (HTML, JavaScript e CSS) che verrà interpretato dal Client.

Ad esempio, una pagina PHP con il seguente codice:

```
<html>
    <head>
    <title>La mia prima pagina php</title>
    </head>
    <body>
    <p>
    <?php echo 'ciao'; ?>
    </p>
    </body>
</html>
```

dopo l'elaborazione del Server restituirà al browser il seguente codice:

```
<html>
    <head>
    <title>Mia prima pagina php</title>
    </head>
    <body>
    <p>
    ciao
    </p>
    </body>
</html>
```

L'aspetto essenziale, ben inteso, è che per il Client sarà impossibile visualizzare il codice sorgente cioè sarà impossibile risalire al codice PHP: l'utente potrà vedere solo codice HTML, JavaScript e CSS.

PHP

PHP, acronimo ricorsivo di PHP: Hypertext PreProcessor (preprocessore di ipertesti) è un linguaggio di programmazione interpretato utilizzato come scripting lato Server per la progettazione di pagine Web dinamiche e applicazioni Web.

Il linguaggio PHP è nato nel 1994, ad opera del programmatore danese Rasmus Lerdorf, con lo scopo di semplificare la creazione delle pagine Web dinamiche. Da sempre il PHP è considerato un progetto open source, cioè liberamente utilizzabile e modificabile da tutti. Questa, probabilmente,

è una delle ragioni che lo ha portato in breve tempo a essere il diretto concorrente del linguaggio closed source della Microsoft, l'ASP.

Oggi il PHP è il linguaggio più utilizzato sul Web. Fra i siti più noti possiamo annoverare:

- Wikipedia
- Wordpress
- Facebook (interfaccia utente, [ref])
- Flickr

Nonostante questo il PHP non è l'unico linguaggio utilizzato. Esistono infatti molte altre tecnologie, per esempio ASP.net, JSP, Ruby on Rails, o Django che per certi aspetti sono anche più avanzate di PHP, e molto apprezzate dagli sviluppatori professionisti. Possiamo però affermare che il PHP è un must in questo campo, e un valido punto di partenza per imparare a mettere mano al Web "da professionisti".

Riferimenti:

http://www.w3schools.com/

http://www.php.net/manual/it/

http://php.html.it/

https://www.mrwebmaster.it

http://www.html.it/guide/guida-php-di-base/

ESERCIZI IN PHP

Nei prossimi esercizi vedremo come creare ed eseguire qualche semplice script in PHP, per prendere confidenza con l'ambiente di lavoro.

ESECUZIONE DI UNO SCRIPT PHP

Sebbene il linguaggio PHP è simile al C (si dice C like) si differenzia dai linguaggi ad alto livello (es, C, Java, Python, etc.), per il fatto che mentre in C il codice che viene scritto (il file sorgente) deve essere prima compilato e successivamente può essere eseguito, il codice realizzato in PHP vive solamente all'interno delle pagine Web.

LA PIATTAFORMA EASYPHP

Abbiamo già parlato di easyPHP, una piattaforma che include tutti gli strumenti necessari a creare pagine Web dinamiche utilizzando la tecnologia PHP e interrogando il DB con MySQL in ambiente Apache.

EasyPHP è uno dei pacchetti più utilizzati in ambiente Windows locale. Wamp Server può essere una valida alternativa.

In ambiente locale Mac OSX, i programmatori utilizzano Mamp, che funziona similarmente al pacchetto per Windows.

ES01 – La mia prima pagina in PHP – Hello World!

Si ricorda che, nella creazione di pagine Web in PHP, i file:

- Devono avere estensione .php.
- Devono essere salvati nella sottocartella \data\localweb del programma easyPHP (si suppone che sia attivo il Web Server locale). In particolare se un progetto è costituito da più file PHP questi devono essere salvati tutti nella stessa cartella (es: ES01) di \data\localweb.
- Devono contenere codice informatico (istruzioni terminanti con il punto e virgola) racchiuso fra il tag di apertura **<?php** e il tag di chiusura **?>**. Sinteticamente, la sintassi PHP è la seguente:

```
<?php
     primaIstruzione;
     secondaIstruzione;
?>
```

- Per poterli eseguire e visualizzarne l'elaborazione occorre aprire il browser e digitare il seguente URL nella barra degli indirizzi:

  localhost/ES01/index.php

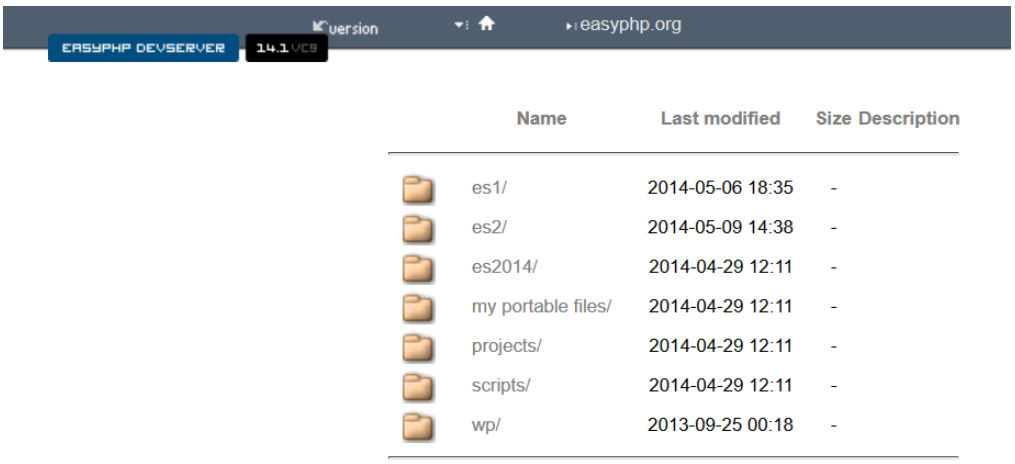
  se il progetto è contenuto nella cartella ES01 di \data\localweb e il file PHP è index.php.

Per realizzare la prima pagina in PHP seguire la seguente procedura:

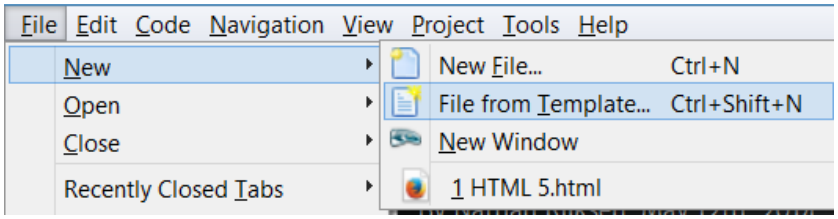
1. Aprire easyPHP.
2. Digitare nella barra degli indirizzi del browser

   localhost

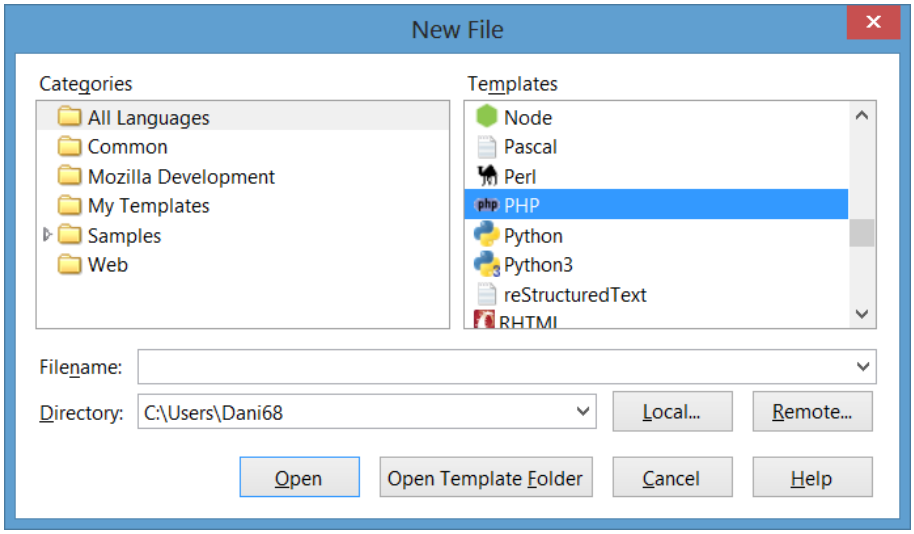
Verranno elencati tutti i progetti in PHP attualmente presenti nella sotto-cartella \data\localweb.



3. Creare in questa sotto-cartella la cartella ES01 (formalmente per ogni esercizio che si realizzerà, occorrerà creare una cartella con questa sintassi ES01, ES02, etc.).
4. In questa cartella ES01 verranno salvati tutti i file del progetto/sito web.
5. Aprire il programma Komodo Edit (o in alternativa Notepad++).
6. Creare un nuovo file PHP in questo modo:
    a. File→New→File from Template…



    b. Selezionare il Template PHP, assegnare un nome al file (es index.php), salvarlo nella cartella ES01 e confermare su Open.



7. Ricopiare il seguente codice:

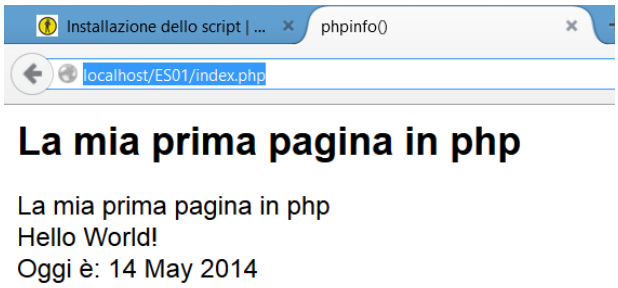
Start Page | index.php

```
<html>
<head><h1>La mia prima pagina in php</h1></head>
<body>

<?php
        echo "La mia prima pagina in php" . "<br />";
        echo "Hello World!";
        echo "<br />";
        echo "Oggi è: ";
        echo Date("d F Y");
?>

</body>
</html>
```

8. Salvare (CTRL+S).
9. Digitare nella barra degli indirizzi il percorso per aprire questa pagina:

http://localhost/ES01/index.php

10. Osservare il risultato ottenuto.



Diamo una breve spiegazione del funzionamento del sistema client-server.

Il nostro sito, http://localhost, cerca i file da visualizzare di default nella cartella LetteraUnità:\...\data\localweb, per cui quando digitiamo http://localhost/ES01/index.php nella barra degli indirizzi del browser avviene quanto segue:

1. L'utente digita l'indirizzo della pagina (es: http://localhost/ES01/index.php) nel browser.
2. Premendo invio il browser "chiama" la macchina (nel nostro esempio localhost), dove Apache è in ascolto per accettare richieste.
3. La richiesta partita dal client dell'utente raggiunge la macchina server che ospita il sito (nel nostro esempio sulla stessa macchina grazie al Web Server Locale che traduce il link http://localhost/ES01/index.php in LetteraUnità:\percorsoEasyPHP\data\localweb), dove appunto Apache è in ascolto per accettare richieste (nel nostro caso localhost).
4. La pagina richiesta è index.php salvata in ES01; Apache sa che deve cercarla nella cartella LetteraUnità:\...\data\localweb /ES01/index.php.

5. Apache, verifica se si tratta di una pagina HTML o php. Essendo una pagina con estensione .php, esegue, riga per riga, il codice PHP in essa contenuta e restituisce il risultato al nostro browser.
6. Il software sul server quindi crea la pagina Web richiesta dinamicamente ("al volo") e la invia al browser del client.
7. Il browser interpreta la pagina HTML e la visualizza all'utente. Il browser quindi non interpreta il codice PHP, anzi, non lo vede ne anche: il browser vede solo codice HTML generato "al volo" dal server.

Analizziamo riga per riga il codice contenuto in questo esempio:

Start Page × index.php ×

```
1  <html>
2  <head><h1>La mia prima pagina in php</h1></head>
3  <body>
4
5  <?php
6          echo "La mia prima pagina in php" . "<br />";
7          echo "Hello World!";
8          echo "<br />";
9          echo "Oggi è: ";
10         echo Date("d F Y");
11 ?>
12
13 </body>
14 </html>
```

È facile riconoscere nel codice la classica struttura di una pagina HTML. Infatti, a parte le righe dalla 5 alla 11, la pagina verrà servita così com'è da Apache.

Il codice scritto invece dalla riga 5 alla 11, verrà eseguito riga per riga dall'interprete PHP; osservate che il codice PHP è scritto tra i blocchi `<?php` e `?>`.

Concentriamoci, quindi, sul codice contenuto in questi due blocchi:

```
5   <?php
6         echo "La mia prima pagina in php" . "<br />";
7         echo "Hello World!";
8         echo "<br />";
9         echo "Oggi è: ";
10        echo Date("d F Y");
11  ?>
```

Le prime quattro righe di codice contengono la stessa istruzioni echo a cui viene passata la stringa di testo da visualizzare a video; in particolare la riga 7 innesta del codice HTML, il tag <br /> utilizzato per mandare a capo il successivo testo.

La quinta riga è sempre una funzione echo, a cui però viene passata a sua volta un'altra funzione: date(). La funzione date() restituisce la data corrente sotto forma di stringa da visualizzare a video.

Osserviamo i parametri passati alla funzione date():"d-m-Y". La chiamata di questa funzione produrrà il giorno del mese su due cifre (d sta per day), un trattino (-), il mese dell'anno su due cifre (m sta per month), un trattino (-) e l'anno su quattro cifre (Y maiuscola sta per year).

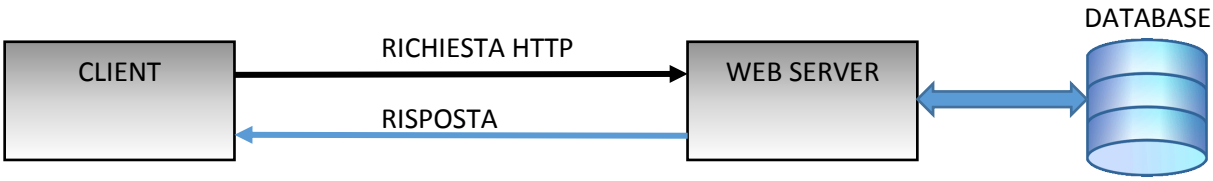
Verificate cosà verrà mostrato a video se alla funzione date() viene passata la stringa "G:i".

Un Web Server è un'applicazione in esecuzione su un computer host (server) che fornire dati o servizi ai client che ne fanno richiesta (il browser del computer dell'utente).

I programmi di Web Server più utilizzati sono:

- Microsofot IIS (Internet Information Services) per ambiente Windows.
- Apache per ambiente Linux, open source.

Il Web Server può essere installato anche sul computer del client.



Il Server Web riceve le richieste di trasferimento di pagine web tramite l'utilizzo del protocollo HTTP (porta 80) o HTTPS (porta 443).

Le richieste vengono effettuate dal client attraverso un Web Browser.

Digitando, quindi, nella barra degli indirizzi del browser l'URL della pagina web che il client vuole richiedere al server, nel nostro caso localhost\ES01\index.php,

Se la richiesta è la visualizzazione di una pagina web statica (es: index.html), il server risponderà semplicemente inviando le informazioni codificate in linguaggio HTML memorizzate nel proprio hard disk. Sarà cura del browser decodificarle al fine di mostrarle correttamente all'utente.

Viceversa se la richiesta è più complessa (es: index.php), per esempio per visualizzare i dati relativi ai voti di un alunno, e parliamo di pagine dinamiche, non è più sufficiente che il server cerchi nel proprio hard disk la pagina desiderata, ma anzi la deve costruire dinamicamente prelevando informazioni da un DataBase attraverso l'utilizzo di tecnologie appropriate. Le più utilizzate sono:

- ASP della Microsoft per ambiente Windows, basato sul linguaggio Visual Basic Script.
- ASPX, che ha sostituito l'ASP, basato sul linguaggio C#.
- PHP per ambiente Linux, basato sul linguaggio C like.
- JSP basato sul linguaggio Java Script.

PHP è una tecnologia che consente di scrivere codice eseguibile (script) lato Server

PHP è un linguaggio di scripting che estende le funzionalità del Web Server consentendo l'interpretazione di file con estensione .php contenenti il codice dell'applicazione.
Il codice viene inserito nelle pagine Web, come già visto nel primo esempio.
Una pagina web PHP è un normale file di testo, salvato con l'estensione .php, che può contenere testo, codice HTML, o script scritti con altri linguaggi (es: JavaScript).
La pagina PHP è eseguita sul Server in risposta a una richiesta proveniente dal browser dell'utente.
Il Server esegue tale script, riga per riga, e restituisce i risultati sotto forma di pagina HTML che viene appunto costruita **dinamicamente** e inviata al browser dell'utente.
Provare a fare clic destro in un qualsiasi punto della pagina Web visualizzata nell'esempio ES01 e scegliere dal menu contestuale la voce che consente la visualizzazione del codice sorgente.
Si otterrà questo:

```
1 <html>
2 <head><h1>La mia prima pagina in php</h1></head>
3 <body>
4
5 La mia prima pagina in php<br />Hello World!<br />Oggi è: 14 May 2014
6 </body>
7 </html>
```

Notate che non si trova nessuna traccia del codice php.
Questo significa che l'utente finale che utilizza il browser non può vedere il sorgente della pagina PHP residente sul server, ma solo la pagina HTML generata dallo script eseguito dal server.

Lo sviluppatore di pagine PHP può verificare il funzionamento degli script sulla propria macchina, come abbiamo fatto nel nostro primo esempio, prima di pubblicarli in Internet, utilizzando un Web Server installato localmente sul suo computer.

Per verificare che il Web Server è configurato correttamente e possa supportare il linguaggio PHP, si può eseguire una pagina PHP costituita dal seguente codice:

ES02

```
<?php
    phpinfo();
?>
```

Se tutto funziona correttamente, viene generata una pagina Web contenente le informazioni sull'interprete PHP installato sul computer: la versione di PHP, le caratteristiche del Web Server e tutte le configurazioni di PHP.
I comandi e le istruzioni del linguaggio PHP vengono scritte all'interno di pagine Web racchiuse da particolari delimitatori. L'inizio del blocco di codice PHP è indicato dalla sequenza di caratteri **<?php** e la fine da **?>**.

ES03

```
<?php
     //   commento su una singola riga
     #    questo è un altro commento su singola riga
     /*   commento su
          più righe
     */
     echo("<br />Pagina dinamica generata con il linguaggio <b>PHP</b>");
?>
```

I commenti sono molto utili al programmatore per descrivere del codice particolarmente importante o complesso a maggior ragione se si lavora in equipe.
Osservate la riga contenente l'ultima istruzione `echo()`. A tale funzione viene passata una stringa tra virgolette come parametro. La stringa può contenere testo e tag HTML. La funzione *echo* invia una stringa al browser dell'utente.

NORME DI BUONA SCRITTURA DEL CODICE

Al fine di abituarsi a utilizzare una **scrittura ordinata** si ricordi di rispettare i seguenti tre criteri:

- Istruzioni su righe separate attraverso l'utilizzo del ritorno a capo (Return).
- L'identazione realizzata con il tasto di tabulazione (TAB) alla sinistra del tasto Q.
- I commenti (su una o più righe).

Se si rispettano questi tre criteri il codice risulterà maggiormente leggibile e chiaro.

Cosa abbiamo visto nell'ES01

Gli script PHP non sono eseguiti dal browser, ma da un server web, che invia il risultato al browser sotto forma di pagina HTML pronta da visualizzare.

Possiamo trasformare il nostro computer in un Web Server installando easyPHP in ambiente Windows.

Apache, il software che gestisce il Web Server, creerà una cartella apposta in cui andremo a mettere le pagine che creiamo. Queste pagine saranno visualizzabili in un qualsiasi browser, all'indirizzo `http://localhost/crtella/pagina.php/`.

Una pagina PHP è una normale pagina HTML, in cui però si trovano dei blocchi di codice, delimitati da `<?php ... ?>`.

Tutto il codice in questi blocchi sarà eseguito come codice PHP.

Possiamo scrivere del testo (es. Hello, world!) nella pagina con l'istruzione echo (es. echo "Hello, world!";).

Attenzione a non dimenticare il punto e virgola al termine di ogni istruzione.

Possiamo chiamare delle funzioni che fanno del lavoro per noi. Ad esempio, la funzione `date()` ritorna la data di oggi, che possiamo stampare con echo (es. echo date("d-m-Y");).

PHP ha un utile manuale che descrive tutte le sue funzioni di base, e come si usano: vale la pena di abituarsi a cercare informazioni nel manuale, visto che PHP conta più di 2000 funzioni predefinite che potrebbero esserci utili in qualsiasi momento, e che sarebbe difficile imparare a memoria.

Riferimenti

http://www.php.net/manual/it/function.echo.php

CODICE PHP INSERITO IN UNA PAGINA HTML

Il codice PHP può convivere con l'HTML (quasi sempre è così) purché sia separato da questo attraverso i tag di apertura e chiusura del PHP. Quindi, se si suppone di avere la seguente pagina Web scritta in HTML:

```
<html>
    <head>
    <title>pagina html</title>
    </head>
    <body>
    <p>Ciao</p>
    </body>
</html>
```

Ebbene il codice PHP potrà essere inserito (a seconda dei casi e delle esigenze) prima, in mezzo o in fondo al codice HTML:

```
<html>
    <head>
    <title>pagina html</title>
    </head>
    <body>
    <p>
        <?php echo "ciao"; ?>
    </p>
    </body>
</html>
```

Inoltre si potrà alternativamente intervallare con l'HTML, come proposto in questo esempio:

```
<html>
    <head>
    <title>
        <?php echo "pagina html"; ?>
    </title>
    </head>
    <body>
    <p>
        <?php echo "ciao"; ?>
```

```
    </p>
    </body>
</html>
```

ES04 – UTILIZZO DELLE VARIABILI

Una variabile è un **contenitore** all'interno del quale possono essere salvati diverse tipologie di dato. Nel linguaggio PHP le variabili sono identificate da un nome preceduto dal simbolo di dollaro (**$**), senza spazi. I nomi delle variabili sono case-sensitive (distinguono maiuscolo e minuscolo) e, per essere validi, devono iniziare con una lettera; possono contenere lettere, cifre e il carattere underscore (_).

Non è necessario dichiarare le variabili, quindi una variabile viene generata nel momento in cui le si assegna un valore.

Si realizzi il seguente esempio:

```
1  <html>
2  <head><h1>Utilizzo delle variabili in php</h1></head>
3  <body>
4
5  <?php
6          $dataDiOggi=date("d-m-Y");
7          echo "Ciao, oggi è il ";
8          echo $dataDiOggi;
9  ?>
10
11 </body>
12 </html>
```

In questo esempio la stringa restituita dalla funzione `date()` viene salvata nella variabile `$dataDiOggi` (riga 8). Questa riga di codice è un'assegnazione.

Questa variabile viene poi riutilizzata nella riga 8 per visualizzare a video il suo contenuto grazie alla funzione echo.

Cosa sono le variabili

Una variabile è un riferimento in memoria RAM a un dato in particolare (un numero, una stringa, etc.) e possiede le seguenti tre proprietà principali: tipo, valore e indirizzo.

- Il **tipo** specifica il dato da rappresentare definendo un range di valori possibili che può assumere una variabile. Ad esempio il tipo int rappresenta un numero intero, mentre un char

un carattere dell'alfabeto, etc. osserviamo che il compilatore distingue i diversi tipi in base alla dimensione della variabile. Quando definiamo un int, viene associata alla variabile una dimensione specifica, che sarà diversa da quella che contiene un carattere.

- Il **valore** è il contenuto presente nella cella di memoria in cui è stato salvato il dato. Quando facciamo un assegnamento a una variabile, ad esempio $numero = 5, riempiamo la porzione di memoria dedicata alla variabile $numero con il valore binario 101, che rappresenta il 5 in decimale. Da osservare che a seconda dell'architettura della macchina su cui eseguiamo il programma, la dimensione di una cella di memoria per un numero intero può essere 32 o 64 bit (32 o 64 "caselline" che possono valere 0 o 1). Nel nostro esempio supponendo di avere celle da 32 bit, prima di 101 ci saranno 29 zeri.
- Ogni variabile ha un **indirizzo** associato, in modo da poter risalire alle informazioni descritte nei precedenti punti. Non è il contesto per descrivere il modo con cui vengono gestitigli indirizzi di memoria.

Osservazione 1

Il linguaggio PHP, diversamente dal C o dal Java non richiede di specificare il tipo delle variabili. L'interprete si occuperà di trattare il tipo della variabile a seconda del suo contenuto. Per questo motivo PHP viene detto un linguaggio a debole tipizzazione.

Osservazione 2

Se una variabile viene letta prima che le sia assegnato un valore, PHP, a seconda del contesto, associa un valore di default. Ad esempio nel caso in cui una variabile non assegnata venga coinvolta in un'operazione aritmetica (es. 3 + $num) il suo valore sarà 0, mentre nel caso di un confronto tra stringhe assumerà il valore della stringa vuota "".

Osservazione 3

Nonostante non si debba dichiarare una variabile in PHP, è sconsigliato dichiararla senza prima avergli assegnato un valore, perché si rischia di scrivere codice corretto sintatticamente (cioè che viene eseguito senza errori), ma non semanticamente (cioè che non ha il significato che vorremmo dargli, e produce quindi risultati sbagliati).

Suggerimento

PHP di default non segnala gli errori meno gravi. Nel file php.ini (il file di configurazione di PHP) è possibile modificare alcuni parametri in modo da ricevere warning in caso di una variabile priva di valore aggiungendo semplicemente la dichiarazione `error_reporting(E_ALL)`. In alternativa è possibile utilizzare tale dichiarazione all'inizio di qualsiasi script: ricordandoci di inserirla tra <?php e ?>, ad esempio:

```
<?php
    $a = $b + 1;
     /*
     A $b non è mai stato assegnato un valore (PHP userà 0 come valore di
     default).
     Questo è considerato un errore non critico (Notice), e non viene
     segnalato.
     */
    error_reporting(E_ALL);
     // Attiva la segnalazione di tutti i livelli di errore
    $a = $c + 1;
     /*
     Stesso errore, ma ora PHP mostra un messaggio Notice sulla pagina
     */
    ?>
```

La debole tipizzazione ha il vantaggio di rendere il codice più snello, più semplice da scrivere ma può nascondere errori semantici difficili da riscontrare soprattutto se siamo alle prime armi con il linguaggio PHP.

Le tipologie di dato contenibili da una variabile sono:

- una stringa;
- un numero;
- un array;
- un valore booleano;
- un oggetto.

NB Lo studio delle tipologie verrà affrontato di volta in volta attraverso degli esempi.

ES 05 - DICHIARAZIONE E ASSEGNAZIONE DI VARIABILI

Come visto nell'esempio iniziale, una variabile in PHP è sempre introdotta dal carattere jolly $, a cui segue il nome che la identifica. Il nome di una variabile è ovviamente scelto dal programmatore; è importante scegliere per ogni variabile un nome che ne rappresenti il contenuto in un colpo d'occhio, in modo da mantenere il codice leggibile. Ad esempio, se devo creare una variabile per contenere la data di oggi, sarà meglio chiamare questa variabile `$dataOggi`, piuttosto che `$pippo` o `$x`.

Per dichiarare una variabile (ovvero crearne una nuova) in PHP è sufficiente usarla:

```
$primaVariabile;
```

PHP crea una nuova variabile quando ne incontra una con un nome mai usato prima.

Per assegnare un valore a una variabile è necessario utilizzare l'operatore di assegnamento = (da non confondere con l'operatore di uguaglianza ==):

```
$primaVariabile = "prima variabile";
```

PHP inoltre consente di poter riassegnare valori di tipi diversi a una variabile precedentemente assegnata:

```
$primaVariabile = "prima variabile"; //Rappresenta una stringa
$primaVariabile = 1; //Rappresenta un intero
```

La ragione di questa possibilità è la stessa che abbiamo spiegato precedentemente. PHP è indulgente sui tipi, quindi è possibile ridefinire un valore di una variabile che precedentemente rappresentava un tipo diverso.

Realizziamo il seguente esercizio:

Start Page | index.php

```
1  <html>
2  <head><h1>Gestione variabili in php</h1></head>
3  <body>
4
5  <?php
6          $contatore=100;
7          $prezzo=14.35;
8          $testo="Pagina dinamica PHP";
9          echo "numero intero: ";
10         echo $contatore;
11         echo "<br />";
12         echo "numero in virgola mobile: ";
13         echo $prezzo;
14         echo "<br />";
15         echo "stringa: ";
16         echo $testo;
17 ?>
18
19 </body>
20 </html>
```

Le variabili assegnate all'interno di un file PHP sono **variabili globali**, cioè sono visibili e possono essere utilizzate in qualsiasi punto del file, anche all'interno di diversi tag <?php … ?>.

Con i numeri e le variabili numeriche si possono utilizzare i seguenti **operatori aritmetici**:

Simbolo	Operatore	Esempio
+	Somma	$z = $y + $x ;
-	Sottrazione	$z = $y - $x ;
*	Moltiplicazione	$z = $y * $x ;
/	Divisione	$z = $y / $x ;
%	Resto della divisione	$z = $y % $x ;
++	Incremento unitario	$x++;
--	Decremento unitario	$x--;

Per assegnare un valore a una variabile, oltre all'operatore di assegnamento (=), esistono questi altri **operatori di assegnamento combinati**, che consentono l'utilizzo del valore di una variabile all'interno di un'espressione senza indicarla esplicitamente:

Operatore	Esempio	Equivalente a
+=	$y += $x ;	$y = $y + $x ;
-=	$y -= $x ;	$y = $y - $x ;
*=	$y *= $x ;	$y = $y * $x ;
/=	$y /= $x ;	$y = $y / $x ;
%=	$y %= $x ;	$y = $y % $x ;
.=	$y .= $x ;	$y = $y . $x ;

Gli operatori di confronto fra due valori sono:

Simbolo	Operatore
==	Uguale a
!=	Diverso da
<	Minore di
>	Maggiore di
<=	Minore o uguale di
>=	Maggiore o uguale di

Gli operatori logici per unire più espressioni sono:

Simbolo	Operatore
&&	And
||	Or
!	Not
xor	Xor

ES04 – LA STRUTTURA DI SELEZIONE E LE FUNZIONI ISSET()

Nei precedenti paragrafi si è accennato alla possibilità di poter dichiarare una variabile senza assegnarli un valore. Per evitare di commettere errori semantici la libreria standard del PHP mette a disposizione una funzione che permette di verificare se una variabile è stata precedentemente definita:

`isset()`

La funzione `isset()` restituisce TRUE se la variabile esiste e quindi si riferisce a un valore, mentre restituisce FALSE in caso contrario.

Realizzare il seguente esempio:

```php
<html>
<head><h1>Selezione e isset</h1></head>
<body>

<?php
        $base = 2;
        $altezza = 5;
        $area = $base * $altezza;
        if(isset($base))          // TRUE
                echo "var base definita";
        else
                echo "var non definita";
        echo "<br />";
        if(isset($altezza))       // TRUE
                echo "var altezza definita";
        else
                echo "var non definita";
        echo "<br />";
        if(isset($area))          // TRUE
                echo "var area definita";
        else
                echo "var perimetro non definita";
        echo "<br />";
        if(isset($perimetro))     // FALSE: la var $perimetro non è stata mai definita
                echo "var definita";
        else
                echo "var non definita";
?>

</body>
</html>
```

ES07 – LA STRUTTURA DI SELEZIONE E LE FUNZIONI EMPTY()

La funzione `empty()` invece controlla il contenuto della variabile. Restituisce TRUE se la variabile che gli viene passata come argomento è NULLA (una stringa vuota, un numero 0, non definita o di valore NULL), FALSE in caso contrario.

Questa funzione è molto utile, per esempio, per verificare se l'utente ha compilato i campi di un form.

Realizzare questo esempio:

```
1  <html>
2  <head><h1>Selezione e isset</h1></head>
3  <body>
4  <form action="index.php" method="get">
5  Inserisci il valore da associare alla variabile (oppure niente)
6  <input type="text" name="funz" size="2" maxlength="2">
7  <input type="submit" value="Verifica"><br />
8  </form>
9  <?php
10         if(isset($_GET['funz']))
11         {
12                 $funz=$_GET['funz'];
13                 $funz2=empty($funz);
14                 echo "Ecco cosa restituisce il controllo della variabile:";
15                 if($funz2==true){echo "true";}
16                 if($funz2==false){echo "false";}
17                 if($funz2==1){echo "1";}
18                 if($funz2==0){echo "0";}
19                 if($funz2==null){echo "Null";}
20         }
21 ?>
22
23 </body>
24 </html>
```

Si analizzerà in dettaglio più avanti il significato del codice scritto. Per il momento è importante soltanto notare il codice dalla riga 12 alla riga 19. Se l'utente non scrive nulla nel campo di testo del form verrà visualizzato **true** altrimenti **false**.

Finora abbiamo visto che...

- Una variabile è uno spazio di memoria in cui possono essere memorizzati dei dati da usare in futuro.
- Ogni variabile ha un nome scelto dal programmatore, e tre caratteristiche
- Tipo - il tipo di dato contenuto nella variabile; ad esempio un numero, una stringa di testo, o un valore booleano (vero o falso).
- Valore - il dato vero e proprio contenuto nella variabile.
- Indirizzo - un codice che dice in quale parte della memoria si trova la variabile. In PHP non ci dobbiamo preoccupare degli indirizzi: sono gestiti automaticamente.
- PHP è un linguaggio a tipizzazione debole, cioè non pone vincoli sul tipo di una variabile. Questo significa che il programmatore non deve dichiarare un tipo per ogni variabile: PHP lo assegnerà (e cambierà, se necessario) automaticamente, a seconda del dato salvato nella variabile.
- Una variabile in PHP è sempre introdotta dal dollaro ($), seguito dal nome della variabile.
- Ogni variabile che viene nominata per la prima volta verrà automaticamente creata da PHP.

- A una variabile può essere assegnato un valore con l'operatore = (es. $pi = 3.14;)
- Le funzioni empty() e isset() servono a controllare se una variabile esiste e se contiene qualcosa.

ES08 – LA GESTIONE DELLE STRINGHE

Continuiamo l'analisi delle variabili parlando di un altro tipo fondamentale del PHP. Il tipo **stringa**. In informatica, in generale, quando si parla di stringhe ci si riferisce genericamente a dei "pezzi" di testo qualsiasi, e in PHP una stringa è esattamente questo: una sequenza di caratteri, come quella che si vede in questo esempio:

```
<?php
    $s = "Ciao, io sono una stringa!";
    echo $s;
?>
```

Si noti che l'istruzione **echo**, che fino ad ora era stata utilizzata per scrivere direttamente il testo da stampare, ora stampa il contenuto della variabile $s. Una variabile stringa (es. $s) infatti sostituisce in tutto e per tutto una costante stringa (es. "Hello, world!"): PHP le tratterà esattamente allo stesso modo.

Stringhe e tipi dinamici

Come abbiamo già visto con le variabili numeriche, una delle caratteristiche del PHP è la capacità di adattare il tipo di una variabile a seconda dell'uso che se ne fa (una caratteristica che in informatica si chiama tipizzazione dinamica). Questo vale anche per le stringhe, e lo possiamo verificare cercando, ad esempio, di stampare un numero intero:

```
<?php
     // Una variabile di tipo 'integer'. PHP riconosce il tipo dal fatto
    $n = 42;
    // che 42 è un numero intero.
    // Una variabile di tipo 'string'. Stavolta PHP riconosce una
    // stringa, perchè "42" è tra virgolette.
    $s = "42";
```

```
    echo $s; // È la normale stampa di una stringa.
    echo $n; // Non dovrebbe funzionare: 'echo' si aspetta una stringa
    // ...invece funziona, perchè PHP adatta i tipi automaticamente!
?>
```

In questo esempio, come vediamo dai commenti, stiamo usando una variabile di tipo intero ($n) come argomento dell'istruzione echo (linea 10). In PHP questo è perfettamente lecito, e si può fare tranquillamente; al contrario, nella maggior parte degli altri linguaggi di programmazione produrrebbe un errore, perché echo si aspetta di ricevere una stringa, e invece si ritrova un numero.

Soffermiamoci un attimo sulla differenza tra i due casi: è normale pensare che non ci sia una gran differenza tra 42 e "42" (con le virgolette). Agli occhi di un computer, però, 42 e "42" sono due cose completamente diverse:

$n = 42 (senza virgolette) è un numero intero. Significa, a grandi linee, che il computer memorizza il corrispondente binario di 42 (101010) in una cella di memoria RAM. Questo "oggetto" memorizzato è un numero vero e proprio, che potrà essere passato così com'è al processore per effettuare operazioni matematiche.

$s = "42" (con le virgolette) è una stringa. Significa che il computer occuperà diverse celle di memoria RAM per memorizzare i singoli caratteri che compongono il numero 42: una cella per il '4' (che è il carattere numero 52 della tabella ASCII, quindi la cella conterrà 110100, 52 in binario) e una per il due (carattere numero 50 della tabella ASCII, quindi 110010 in binario). Questi codici che vengono memorizzati non hanno nessun senso dal punto di vista numerico, ma servono solo a identificare i caratteri che compongono il numero 42.

Abbiamo visto quindi che le due variabili $n e $s dell'esempio contengono due cose completamente diverse se andiamo a guardare cosa succede in memoria ($n memorizza 00101010, $s memorizza 00110100 00110010). Come però abbiamo già accennato, questo discorso lo facciamo ora per capire la differenza tra i due tipi di dato, ma non ci servirà più, almeno in questa guida: PHP infatti è fatto per convertire automaticamente le variabili da un tipo all'altro quando è necessario, in modo che il programmatore non se ne debba preoccupare.

Questo l'abbiamo visto nell'esempio qui sopra, dove abbiamo stampato un numero intero come se fosse una stringa, ma naturalmente vale anche il contrario: possiamo fare delle operazioni matematiche con le stringhe, trattandole quindi come se fossero numeri:

```
<?php
      $s = "42";        // Una variabile di tipo 'string'.
      // A questo punto PHP converte $s in un numero intero per
      // effettuare l'operazione.
      $n = $s*2;
      // Stampa 'integer', perché $n è a tutti gli effetti
      // un numero (42*2, ovvero 84)
      echo gettype($n);
  ?>
```

Definire una stringa

I doppi apici, che abbiamo utilizzato finora, non sono in realtà l'unico modo per definire le stringhe. PHP mette a disposizione quattro modi diversi per raggiungere lo stesso obbiettivo:

- doppi apici
- apici singoli
- sintassi "heredoc"
- sintassi "nowdoc"

Doppi apici

Abbiamo già visto come usare i doppi apici per definire una stringa in PHP:

```
<?php
    echo "Hello, world!";
?>
```

Non abbiamo però parlato di alcune particolari caratteristiche di questa sintassi. Ad esempio, due o più stringhe possono essere concatenate usando il carattere punto (.), come mostrato nell'esempio seguente:

```
<?php
    $s1 = "porta";
```

```
    $s2 = "matite";

    $s  = $s1.$s2;     // $s contiene 'portamatite'
?>
```

La concatenazione, come tutte le altre operazioni sulle stringhe, può essere effettuata, oltre che sulle variabili, anche sulle costanti; questo può servire per inserire variabili in mezzo al testo, o per "spezzare" linee di testo troppo lunghe:

```
<?php
    $s  = "una variabile";
    echo "In mezzo al testo inserisco ".$s;
    // stampa: 'In mezzo al testo inserisco una variabile'
    echo "Questa linea di testo è troppo lunga... ".
         "quindi la spezzo in due!";
    /* stampa: 'Questa linea di testo è troppo lunga... quindi la
    spezzo in due!' */
?>
```

Una particolarità dei doppi apici è la possibilità di inserire delle variabili direttamente nel testo, senza bisogno di usare la concatenazione:

```
<?php
    $s  = "una variabile";
    // Variabile inserita con la concatenazione
    echo "In mezzo al testo inserisco ".$s;
    // Stesso risultato, senza concatenazione
    echo "In mezzo al testo inserisco $s";
?>
```

Ogni volta che inseriamo il simbolo del dollaro ($) in mezzo al testo, quindi, PHP andrà a cercare la variabile corrispondente ai caratteri che seguono il dollaro. A volte PHP non può distinguere correttamente il nome di una variabile; in questo caso si usano le parentesi graffe, come mostrato in questo esempio:

```
<?php
    $s = "tre";

    echo "Trenta$s"            // Ok: stampa 'Trentatre'
    echo "Trenta$s $sntini";   // Non funziona: PHP cerca una
                               // variabile di nome $sntini...
    echo "Trenta$s {$s}ntini"; // Ok: stampa 'Trentatre trentini'
    echo "Trenta$s ${s}ntini"; // Ok: stampa 'Trentatre trentini'
?>
```

L'ultima caratteristica interessante dei doppi apici riguarda i caratteri speciali. Infatti, esistono specifiche combinazioni di caratteri (dette anche sequenze di escape) che iniziano sempre con un back-slash (\), e permettono di inserire simboli particolari nel testo. La seguente tabella mostra i più importanti (l'elenco completo si trova, come al solito, nel manuale di PHP):

Sequenza di escape Significato

\n A capo. Nota: gli 'a capo' riguardano il codice HTML della pagina che PHP sta producendo; sono importanti per rendere leggibile il codice HTML, ma non dimentichiamo che non verranno visualizzati nel browser (per quello dovremmo usare appositi tag HTML come, ad esempio, <br />).

\" Stampa i doppi apici. Notare che senza il back-slash i doppi apici verrebbero scambiati per la fine della stringa (es. echo "Ho incontrato Gianni e gli ho detto "Ciao!""; produce un errore, perchè PHP scambia l'apertura delle virgolette, prima della parola Ciao, per la fine della stringa; al contrario, echo "Ho incontrato Gianni e gli ho detto \"Ciao!\""; funziona correttamente).

\$ Stampa il segno del dollaro. Notare che senza il back-slash il dollaro verrebbe scambiato per l'inizio di una variabile, come abbiamo visto nel paragrafo precedente.

\\ Stampa un back-slash.

Apici singoli

In PHP le stringhe possono anche essere delimitate da apici singoli:

```
<?php
    echo "Hello, world!";  // Stampa di una stringa con apici doppi
    echo 'Hello, world!';  // Stesso risultato, ma con apici singoli.
?>
```

Anche se apici singoli e apici doppi si usano esattamente allo stesso modo, le due notazioni non sono del tutto equivalenti: possiamo vedere gli apici singoli come una versione ridotta di quelli doppi. Infatti, gli apici singoli non permettono di inserire variabili o caratteri speciali nel testo: il contenuto della stringa viene stampato così com'è.

Le stringhe tra apici singoli, quindi, sono svantaggiose quando si tratta di usare variabili e caratteri speciali, ma hanno anche un lato positivo: sono più veloci, perché l'interprete PHP stampa la stringa così com'è, e quindi non deve perdere tempo a cercare variabili e caratteri speciali nel testo (in gergo si dice che non viene effettuato il parsing della stringa).

È importante ricordare che la concatenazione con gli apici singoli funziona esattamente come abbiamo visto per gli apici doppi. Inserire una variabile nel testo è sempre possibile, quindi, a patto di usare la concatenazione:

```
<?php
    $s = "una variabile";
    echo 'Inserisco '.$s.' tra apici singoli';
    // Stampa: Inserisco una variabile tra apici singoli
    echo 'Inserisco $s tra apici singoli';
    // Stampa: Inserisco $s tra apici singoli
    // PHP non riconosce la variabile $s. La riconoscerebbe, se fosse
    // tra doppi apici.
?>
```

Inoltre, non è del tutto vero che non esistono caratteri speciali tra apici singoli: ne esiste uno, l'apice singolo stesso.

Sequenza di escape	Significato
\'	Un apice singolo.

Questa sequenza di escape è essenziale quando vogliamo inserire un apostrofo nel testo. Ad esempio, il seguente codice non è corretto:

```
<?php
    echo 'Quando serve, uso l'apostrofo';
```

```
    // Errore! PHP interpreta l'apostrofo
    // come la chiusura della stringa.
?>
```

L'errore può essere facilmente corretto inserendo un back-slash prima dell'apostrofo:

```
<?php
    echo 'Quando serve, uso l\'apostrofo';
    // Ok: il back-slash fa sì che
    // PHP tratta l'apostrofo come un qualsiasi altro carattere.
?>
```

Sintassi "heredoc"

Sebbene apici e doppi apici siano i modi più comunemente usati per delimitare le stringhe in PHP, questo linguaggio ne mette a disposizione di altri. Un esempio è la sintassi heredoc, che risulta comoda quando il testo da scrivere si estende su più righe:

```
<?php
    $s = <<<EOT
In questo spazio posso
scrivere quello che voglio
andando a capo, se serve...
EOT;

    echo $s;
?>
```

La sintassi heredoc (dall'inglese Here document) è nata nel mondo Unix qualche decina di anni fa, per consentire ai programmatori di inserire in libertà lunghi frammenti di testo nel loro codice. Cerchiamo di capire come funziona:

La stringa è aperta alla linea 2. Notiamo che, al posto dei soliti apici, si trova il codice <<<EOT

<<< è il "segnale" grazie a cui l'interprete PHP capisce che sta per essere definita una stringa heredoc.

EOT è un identificatore che può essere scelto a piacimento dal programmatore. Questo stesso codice dovrà essere usato per chiudere la stringa. Nell'esempio è stato usato il codice EOT (che sta per End Of Transmissions, "Fine delle trasmissioni" in italiano); questo dice all'interprete di trattare tutto il testo che segue come una stringa, finché non viene incontrato di nuovo il codice EOT (il che avverrà alla linea 6).

Dopo l'apertura di una stringa heredoc, è necessario andare a capo.

Le linee 3, 4 e 5 contengono il testo vero e proprio che sarà memorizzato nella variabile $s.

La linea 6, come abbiamo accennato, è la chiusura della stringa. Le cose da notare sono due:

Il codice di chiusura può essere scelto a piacere, a patto che, come abbiamo detto, sia lo stesso usato nell'apertura della stringa (linea 2, nell'esempio).

Il codice di chiusura (EOF, nell'esempio) deve essere posizionato all'inizio della riga. Nessun carattere può precedere il codice, nemmeno spazi bianchi o tabulazioni.

La linea 8 contiene una normale echo: la stringa $s viene stampata come qualsiasi altra stringa.

Il testo contenuto in una stringa heredoc si comporta esattamente come il testo tra doppi apici: le variabili contenute al suo interno sono sostituite col valore in esse contenuto, ed è possibile includere gli stessi caratteri speciali di cui abbiamo parlato nel paragrafo Doppi apici e che sono descritti nel manuale. Oltre a questo, in una stringa heredoc possiamo usare tranquillamente i doppi apici senza usare la sequenza di escape \" (che abbiamo incontrato in Doppi apici), perché non rischiano più di essere scambiati per la chiusura della stringa.

```
<?php
    $sito = 'RePNI';
    $indirizzo = 'http://www.repni.it/';

    $testo = <<<REPNIROCKS
<p>Il mio sito preferito è <span style="color:red;">$sito</span>, clicca su <a href="$indirizzo">questo link</a> per visitarlo!</p>
<p>Su $sito ho imparato che in una stringa heredoc posso inserire variabili e doppi apici (").</p>
<p>Utilizzando le sequenze di escape, posso anche inserire caratteri speciali come il dollaro (\$) o il back-slash (\\).</p>
REPNIROCKS;
```

```
    echo $testo;
?>
```

Questo codice produrrà il seguente risultato quando aperto nel browser:

```
Il mio sito preferito è RePNI, clicca su questo link per visitarlo!

Su RePNI ho imparato che in una stringa heredoc posso inserire variabili
e doppi apici (").

Utilizzando le sequenze di escape, posso anche inserire caratteri speciali
come il dollaro ($) o il back-slash (\).
```

Notare che tutti i nomi di variabile sono stati sostituiti con le variabili stesse (es. $sito è stato sostituito con RePNI), i doppi apici sono stati stampati come qualsiasi altro carattere (quelli alla riga 6 chiaramente non si vedono perché sono parte dell'HTML necessario per definire il link), e i caratteri preceduti da back-slash sono stati riconosciuti e stampati come caratteri speciali.

Si potrebbe dire ancora qualcos'altro su heredoc, ma si tratta di sottigliezze che per il momento non ci interessano. Chi fosse curioso può sempre approfondire leggendo la pagina di manuale corrispondente.

Sintassi "nowdoc"

Così come le stringhe a doppi apici hanno un corrispettivo "light", anche la sintassi heredoc ha una versione più leggera, ossia la sintassi newdoc.

Una stringa newdoc è definita in modo quasi identico ad una stringa heredoc:

```
<?php
      $variabile = 'Un testo che non sarà usato...';

      $testo = <<<'EOT'
  Questa è una stringa newdoc.
  All'apparenza è simile a heredoc, ma ha meno funzionalità
```

```
infatti se scrivo $variabile, leggo semplicemente $variabile...
EOT;

    echo $testo;
?>
```

Notiamo subito che l'unica differenza con heredoc sta nel fatto che l'identificatore di chiusura (EOT, alla linea 4) è qui racchiuso tra apici singoli.

Per il resto, il comportamento differisce solo nel fatto che, in nowdoc, PHP non "perde tempo" a cercare variabili e caratteri speciali nel testo, ma stampa tutto così com'è; in gergo si dice che non viene effettuato il parsing della stringa. Ad esempio, alla linea 7 troviamo $variabile nella stringa; Se la stringa fosse stata definita con heredoc (cioè senza apici singoli attorno a EOT) $variabile sarebbe riconosciuto come un identificatore di variabile, e quindi sostituito con il contenuto di $variabile (definito alla linea 2); invece, proprio come avverrebbe se usassimo gli apici singoli, l'output del programma sarà semplicemente $variabile.

In questo capitolo abbiamo visto che...

- Con il termine stringa in informatica ci si riferisce ad una qualsiasi sequenza di caratteri, come ad esempio un frammento di testo.
- In PHP una variabile può essere di tipo stringa, ovvero contenere del testo.
- Grazie alla tipizzazione dinamica, in PHP non ci dobbiamo preoccupare dei tipi di variabile, perché essi, quando possibile, sono adattati automaticamente a seconda della necessità.
- Una stringa può essere definita in quattro modi: apici doppi, apici singoli, heredoc e nowdoc
- Una stringa tra apici doppi espande automaticamente le variabili contenute al suo interno, e riconosce un certo numero di caratteri speciali
- Una stringa tra apici singoli è più veloce da elaborare, ma non espande il contenuto delle variabili e non gestisce altri caratteri speciali che non gli apici singoli stessi.
- Una stringa heredoc si comporta come una stringa tra doppi apici, ma è più comoda da usare quando il testo occupa più righe, non necessita l'escape dei doppi apici.
- Una stringa nowdoc è il corrispettivo delle stringhe tra apici singoli rispetto a heredoc.

http://www.miniscript.it/articoli/21/i_tipi_di_variabili_del_php_le_stringhe.html

ECHO E PRINT, L'OUTPUT DELLE STRIGHE

Dopo aver illustrato come definire le stringhe, vediamo ora come visualizzarle. PHP prevede diverse funzioni per l'output, la principale è **echo()**. In realtà non è una funzione vera e propria, ma un costrutto del linguaggio e quindi possiamo omettere le parentesi tonde.

```
<?php
echo "Consulta il sito html.it";

// la stringa può essere suddivisa su più linee
echo "La stringa si compone
di più linee
ma verrà correttamente stampata";

// si possono inserire dei caratteri new line \n all'interno della stringa
echo "La stringa contiene caratteri new line\ne viene stampata\nsu più linee\ncontrollate il sorgente html";

echo "La stringhe ", "possono essere più di una ";

$stringa = "stringa di prova";
echo $stringa;
echo "Ecco una $stringa";
?>
```

`Echo` prevede una sintassi abbreviata che funziona solo se nel file di configurazione php.ini la direttiva **short_open_tag** è impostata su `on`:

```
<!-- così viene stampata la variabile $stringa -->
<?=$stringa?>
```

La funzione **print()**, anch'essa prevista dal linguaggio, si può considerare praticamente equivalente ad `echo`. Alcuni test hanno però dimostrato che `echo` risulta più veloce del `5-10%` a seconda dei contesti. Si osservi che la sintassi di `print()` non permette argomenti multipli.

```
<?php
$stringa = "stringa di prova";

print "Ecco una $stringa";

// così non funziona
print "La stringhe ", "non possono essere più di una ";
?>
```

FORMATTARE LE STRINGHE: PRINTF

La funzione **printf()** (come l'omologa **sprintf()**), viene utilizzata per produrre un output formattato. È sconsigliabile in termini di prestazioni rispetto alle altre due, va quindi utilizzata solo quando sia effettivamente necessaria.

Si pensi di dover stampare il numero `1/6`: utilizzando `echo` o `print` si otterrebbe il risultato seguente `0.166666666667`. Per ottenere un risultato più leggibile, con due sole cifre dopo la virgola dovremo ricorrere a `printf()`;

```
<?php
// stampiamo (1/6) come 0.17
printf("%1.2f", (1/6));
?>
```

In pratica il primo argomento della funzione **specifica il formato** con cui rappresentare il numero che compare come secondo argomento. Nell'esempio mostrato il numero da stampare dovrà essere costituito dal almeno un carattere, dovrà avere due cifre dopo la virgola e dovrà essere trattato come un numero **floating-point** (`f`).

Forse l'utilizzo della funzione non è immediato, sinteticamente per indicare come formattare l'output si specificano nell'ordine:

- uno o più caratteri (escluso %) che precedono il risultato
- il simbolo %
- uno specificatore di padding (opzionale)
- uno specificatore di allineamento (opzionale)
- uno specificatore di numero minimo di caratteri (opzionale)
- uno specificatore di precisione (opzionale)
- uno specificatore di tipo che indica se l'argomento da stampare sia da considerarsi intero, float, stringa etc.

Per maggiori dettagli si rimanda alla documentazione ufficiale, mentre di seguito si riportano alcuni esempi significativi:

```
<?php
// stampiamo "valore pari a euro 30.25"
printf("valore pari a euro %2.2f", 30.25);

// stampiamo 2 come 00002
printf("%05d", 2);

// stampiamo 2 come ***2
```

```
printf("%'*4d", 2);
?>
```

PHP fornisce anche altre funzioni di output che, per ragioni di spazio e particolarità di utilizzo, non prenderemo in considerazione. Si descriverà più avanti la funzione `number_format()`, utile per la formattazione dei numeri, che permette di evitare il ricorso alla più complessa `printf()`.

Riferimenti:

http://www.html.it/articoli/tutto-sulle-stringhe-in-php-2/

ITERAZIONI – CICLO FOR – ES09

Il ciclo FOR si utilizza quando si devono eseguire per un numero prefissato di volte le stesse istruzioni. Il seguente codice consente il calcolo del fattoriale di un numero intero positivo.

```
<?php
      $n=4;
      $fat=1;
      for($i=1; $i<=$n; $i++){
            $fat=$fat * $i;
      }
    echo "Il fattoriale di " . $n . " e' " . $fat;
?>
```

Questo è il risultato a video:

Ciclo FOR in php

Il fattoriale di 4 e' 24

Rivedremo meglio il ciclo FOR quando introdurremo gli array.

ITERAZIONI – CICLO WHILE – E LE COSTANTI - ES10

Attraverso la funzione rand() si generino 10 voti interi casuali e se ne visualizzi la media in forma tabellare. Il primo metodo fa uso ancora del ciclo FOR, mentre il secondo del ciclo WHILE, ciclo a condizione iniziale.

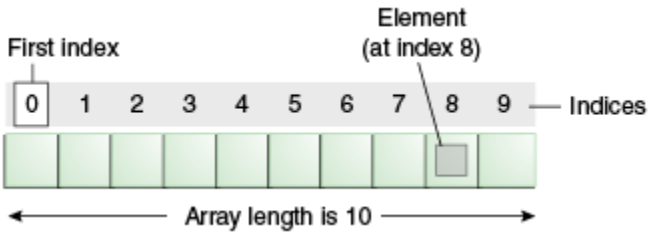
```php
<?php
      $somma = 0;
      echo("<table border=\"1\">");
      echo("<tr>");
      echo("<td>");
      echo("i");
      echo("</td>");
      echo("<td>");
      echo("Voto");
      echo("</td>");
      echo("</tr>");
      for($i = 0; $i < NUM; $i++)
      {
         $voto = rand(1, 10);
         echo("<tr>");
         echo("<td>");
         echo($i);
         echo("</td>");
         echo("<td>");
         echo($voto);
         echo("</td>");
         echo("</tr>");
         $somma = $somma + $voto;
      }
      $media = $somma / NUM;
      echo("</table border=\"1\">");
      echo "<br />" . "MEDIA: " . $media. "<br />";

      $i = 0;
      $somma = 0;
      define('NUM', 10);
      echo("<table border=\"1\">");
      echo("<tr>");
      while($i < NUM)
      {
         $voto = rand(1, 10);
         $somma = $somma + $voto;
         $i++;
```

```
            echo("<td>");
            echo($voto);
            echo("</td>");
        }
        echo("</tr>");
        echo("</table border=\"1\">");
        $media = $somma / NUM;
        echo("MEDIA: " . $media. "<br />");
?>
```

Si noti l'utilizzo della funzione `define` che permette di definire delle costanti. Nel nostro esempio con `define('NUM', 10)` si dichiara una costante numerica NUM pari a 10.
I nomi che si assegnano alle costanti, diversamente dalle variabili, non sono preceduti dal simbolo di dollaro ($). Inoltre, si preferisce, usare nomi maiuscoli, sempre per differenziarle dalle variabili.

VETTORI E CICLO FOREACH – ES11

Gli array sono vettori di variabili. Gli array sono strutture utilizzate per memorizzare elementi accessibili attraverso un indice. L'indice genericamente è un intero compreso fra 0 e N-1 dove N è la dimensione dell'array (il numero di elementi memorizzabili).



Per fare un paragone, se una variabile è un contenitore, allora, un array è un multicontenitore. Gli array monodimensionali sono paragonabili ai vettori matematici, e gli array a 2 dimensioni alle matrici matematiche. In php gli elementi di un array non necessariamente devono essere dello stesso tipo (interi, reali, stringhe, etc.); il primo elemento quindi può essere un intero, il secondo una stringa, il terzo un'immagine, e così via.

Nel esempio che segue si salvano nel vettore $voti 10 voti interi generati casualmente.

```
<?php
	$n = 10;
	for($i=0; $i<$n;$i++){
		$voti[$i] = rand(1, 10);
	}
?>
```

La funzione rand genera un numero intero compreso fra un minimo (il primo parametro) e un massimo (il secondo parametro):

```
rand(min, max)
```

In quest'altro esempio invece si inizializza un array con dei nomi di persona e attraverso un ciclo iterativo si stampano a video:

```
<?php
	echo "ARRAY" . '<br />';
	$nomi=array('daniele', 'marco', 'luigi', 'sergio');
	$n=count($nomi);
	echo "CICLO FOR" . '<br />';
	for($i=0; $i<$n;$i++){
		echo $nomi[$i] . '<br />';
	}
	echo '<br />' . "CICLO FOREACH" . '<br />';
	foreach($nomi as $nome){
		echo $nome . '<br />';
	}
?>
```

La dichiarazione di un array …

```
$nomi=array('daniele', 'marco', 'luigi', 'sergio');
```

Ad ogni elemento dell'array è associato un indice numerico, a partire da 0. Quindi, nel nostro esempio, a `'daniele'` viene assegnato l'indice 0, a `'marco'` l'indice 1, e così via fino a `'sergio'` che avrà indice 3.

Per accedere ad un elemento dell'array si indica il nome dell'array seguito dall'indice contenuto fra parentesi quadre. Per esempio, se si vuole visualizzare il contenuto della seconda cella dell'array:

```
echo $nomi[1]; //stampa 'marco'
```

Se invece si vuole sovrascrivere un dato elemento:

```
$ nomi[3] = 'paolo';
```

L'elemento di indice 3, `'sergio'`, viene sostituito da `'paolo'`.

Per aggiungere altri elementi all'array, e quindi aumentarne la sua dimensione, basta assegnare un valore all'array che deve essere scritto senza indicare l'indice nelle parentesi quadre. Per esempio per aggiungere al precedente array il nome `'aldo'`, si scrive:

```
$nomi[]='aldo';
```

L'array a questo punto avrà dimensione 5.

È molto utile la funzione `print_r` che permette di stampare la struttura di un array. Per esempio:

```
print_r($nomi);
```

E si ottiene:

```
Array ( [0] => daniele [1] => marco [2] => luigi [3] => sergio [4] => aldo )
```

ARRAY ASSOCIATIVI

Gli indici degli elementi non sono solo numerici, ma possono essere anche delle stringhe:

```
$persona['nome'] = 'Sabina';
```

Con questa istruzione abbiamo definito un array di nome $persona, creando un elemento la cui chiave è `'nome'` e il cui valore è `'Sabina'`. È da ricordare che le chiavi numeriche ed associative possono coesistere nello stesso array. Per esempio, salviamo la formazione della squadra di calcio campione del Mondo 1982 in Spagna:

```
$formazione[1] ='Zoff';
$formazione[2] ='Collovati';
$formazione[3] ='Scirea';
$formazione[4] ='Gentile';
$formazione[5] ='Bergomi';
$formazione[6] ='Tardelli';
$formazione[7] ='Oriali';
$formazione[8] ='Cabrini';
$formazione[9] ='Conti';
$formazione[10] ='Rossi';
```

```
$formazione[11] ='Graziani';
$formazione['ct'] = 'Bearzot';
```

In questo caso abbiamo creato un array con dodici elementi, di cui undici con chiavi numeriche, ed uno con chiave associativa. Se in seguito volessimo aggiungere un elemento usando le parentesi quadre vuote, il nuovo elemento prenderà l'indice 12. Avremmo potuto creare lo stesso array usando l'istruzione di dichiarazione dell'array, così:

```
$formazione = array(1 => 'Zoff', 'Collovati', 'Scirea', 'Gentile',
'Bergomi', 'Tardelli', 'Oriali', 'Cabrini', 'Conti', 'Rossi', 'Graziani',
'ct' => 'Bearzot');
```

Analizziamo il formato di questa istruzione: per prima cosa abbiamo creato il primo elemento, assegnandogli esplicitamente la chiave 1. Come possiamo vedere, il sistema per fare ciò è di indicare la chiave, seguita dal simbolo => e dal valore dell'elemento. Se non avessimo indicato 1 come indice, PHP avrebbe assegnato al primo elemento l'indice 0. Per gli elementi successivi, ci siamo limitati ad elencare i valori, in quanto PHP, per ciascuno di essi, crea la chiave numerica aumentando di 1 la più alta già esistente. Quindi `'Collovati'` prende l'indice 2, `'Scirea'` il 3 e così via. Arrivati all'ultimo elemento, siccome vogliamo assegnargli una chiave associativa, siamo obbligati ad indicarla esplicitamente.

È da notare che quando abbiamo usato le chiavi associative le abbiamo indicate fra apici: ciò è necessario per mantenere la 'pulizia' del codice, in quanto, se non usassimo gli apici, PHP genererebbe un errore di tipo 'notice', anche se lo script funzionerebbe ugualmente (dato che il valore verrebbe convertito automaticamente in una stringa). Vediamo ora qualche esempio di creazione e stampa dei valori di un array:

```
$persona['nome'] = 'Mario'; //corretto
$persona[cognome] = 'Rossi'; /*funziona, ma genera un errore 'notice'*/
echo $persona['cognome']; //stampa 'Rossi': corretto
echo "ciao $persona[nome]"; /*stampa 'ciao Mario': corretto (niente apici
fra virgolette)*/
echo "ciao $persona['nome']"; //NON FUNZIONA, GENERA ERRORE
echo "ciao {$persona['nome']}"; /*corretto: per usare gli apici fra
virgolette dobbiamo comprendere il tutto fra parentesi graffe*/
echo "ciao " . $persona['nome']; /*corretto: come alternativa, usiamo il
punto per concatenare (v. lez.10 sugli operatori)*/
```

Abbiamo così visto in quale maniera possiamo creare ed assegnare valori agli array, usando indici numerici o associativi, impostando esplicitamente le chiavi o lasciando che sia PHP ad occuparsene.

ARRAY MULTIDIMENSIONALI

Vediamo ora in che modo possiamo creare strutture complesse di dati, attraverso gli array a più dimensioni.

Un array a più dimensioni è un array nel quale uno o più elementi sono degli array a loro volta. Supponiamo di voler raccogliere in un array i dati anagrafici di più persone: per ogni persona registreremo nome, cognome, data di nascita e città di residenza:

```
$persone = array(
     array(     'nome'          => 'Mario',
                'cognome'       => 'Rossi',
                'data_nascita'  => '1973/06/15',
                'residenza'     => 'Roma'
     ),
     array(     'nome'          => 'Paolo',
                'cognome'       => 'Bianchi',
                'data_nascita'  => '1968/04/05',
                'residenza'     => 'Torino'
     ),
     array(     'nome'          => 'Luca',
                'cognome'       => 'Verdi',
                'data_nascita'  => '1964/11/26',
                'residenza'     => 'Napoli'
     )
);

print $persone[0]['cognome']; // stampa 'Rossi'
print $persone[1]['residenza']; // stampa 'Torino'
print $persone[2]['nome']; // stampa 'Luca'
```

Con questo codice abbiamo definito un array formato a sua volta da tre array, che sono stati elencati separati da virgole, per cui ciascuno di essi ha ricevuto l'indice numerico a partire da 0. All'interno

dei singoli array, invece, tutte le chiavi sono state indicate come associative. Da notare che, sebbene in questo caso ciascuno dei tre array 'interni' abbia la stessa struttura, in realtà è possibile dare a ciascun array una struttura autonoma.
Vediamo un altro esempio:

```
$persone = array( 1 => array('nome' => 'Mario Rossi', 'residenza' =>
'Roma', 'ruolo' => 'impiegato'), 2 => array('nome' => 'Paolo Bianchi',
'data_nascita' => '1968/04/05', 'residenza' => 'Torino'),
'totale_elementi' => 2);
print $persone[1]['residenza']; // stampa 'Roma'
print $persone['totale_elementi']; // stampa '2'
```

In questo caso il nostro array è formato da due array, ai quali abbiamo assegnato gli indici 1 e 2, e da un terzo elemento, che non è un array ma una variabile intera, con chiave associativa 'totale_elementi'. I due array che costituiscono i primi due elementi hanno una struttura diversa: mentre il primo è formato dagli elementi 'nome', 'residenza' e 'ruolo', il secondo è formato da 'nome', 'data_nascita' e 'residenza'.

FUNZIONI

Una funzione è una procedura che a partire da zero, uno o più input, esegue determinate istruzioni al fine di pervenire alla risoluzione di un sottoproblema (elaborazione dati) ed eventualmente nel comunicarne il risultato al chiamante (codice che ha richiamato tale procedura).
Gli input di una funzione sono detti **argomenti** o **parametri** e possono essere facoltativi o obbligatori.
L'output restituito dalla funzione è detto **return** o **risultato**.
La sintassi per la chiamata di una funzione è la seguente:

```
$risultato=nome_Funzione(par1, par2, …, parN);
```

Come già detto il numero di parametri può andare da 0 a N e possono anche essere di tipologia diversa fra loro, per esempio uno intero, un altro decimale, un altro booleano e, un altro ancora stringa.
Le funzioni, a seconda che siano realizzate dall'utente o predefinite del linguaggio, si classificano in:

- **Funzioni native**: PHP dispone di una vastissima serie di funzioni utilizzabili all'occorrenza.
- **Funzioni utente**: sono quelle definite dal programmatore.

FUNZIONI DEFINITE DALL'UTENTE E CARATTERI SPECIALI – ES12

La sintassi per definire una funzione è la seguente:

```
nome_Funzione(par1, par2, …, parN){
   //istruzioni da eseguire con i parametri di input;
   return risultato_della_funzione;
}
```

Facciamo un esempio:

Si vuole realizzare una funzione che calcola le radici di un'equazione di secondo grado. Alla funzione vengono passati i tre coefficienti dell'equazione.

Osserviamo nella zona <head> abbiamo introdotto un meta tag per gestire correttamente i caratteri accentati e speciali attraverso il `charset UTF-8`.

In questa prima versione la funzione non restituisce nulla.

```
<html>
<head><title>array</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<?php
      //prima definisco la funzione
      function eq2($a, $b, $c){
            if($a != 0) {
                  $discriminante = $b * $b - 4 * $a * $c;
                  if($discriminante >= 0) {
                        $x1 = (-$b+sqrt($discriminante))/(2*$a);
                        echo "Prima soluzione: " . $x1 . "<br />";
                        $x2 = (-$b-sqrt($discriminante))/(2*$a);
                        echo "Seconda soluzione: " . $x2 . "<br />";
                        if($a>0) {
                              echo "La parabola è concava";
```

```
                    }
                    elseif($a<0) {
                        echo "La parabola è convessa";
                    }
                }
                else {
                    echo "soluzioni non reali ma complesse";
                }
            }
            else {
                echo "Non è una equazione di secondo grado";
            }
        }

        //richiamo la funzione
        eq2(1, 1, 0);
?>
</body>
</html>
```

Vediamo come sia possibile modificare la precedente funzione affinché restituisca i risultati al chiamante. Osserviamo che con la parola chiave `return` è possibile restituire un solo risultato.

```
<html>
<head><title>array</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<?php
    function eq2_Ver2($a, $b, $c){
        $delta = $b * $b - 4 * $a * $c;
        if($delta >= 0) {
            $x1 = (-$b+sqrt($delta))/(2*$a);
            $x2 = (-$b-sqrt($delta))/(2*$a);
            $delta .=" positivo o uguale a 0";
            $sol = "x1=" . $x1 . "?x2=" . $x2;
        }
```

```
            else {
                  $sol = "non esiste";
            }
            return($sol);
      }
      //utilizzo metodo POST solo se i parametri vengono passati dal form
      /*
      $a = $_POST['a'];
      $b = $_POST['b'];
      $c = $_POST['c'];
      */
      //altrimenti testo assegnando dei valori di esempio
      $a = 1;
      $b = 1;
      $c = 0;

      //richiamo la funzione
      echo "<br />" . eq2_Ver2(1, 1, 0);
?>

</body>
</html>
```

La funzione restituisce il parametro `$sol`, del tipo stringa, contenente le soluzioni (se esistenti) dell'equazione. Successivamente sarà possibile ricavare dalla stringa i valori numerici `$x1` e `$x2` attraverso funzioni native di PHP.

```
$ris=eq2_Ver2($a, $b, $c);
if(strpos($ris, '?')>0){
      $x1=substr($ris, strpos($ris, '=')+1, strpos($ris, '?')-strpos($ris, '=')-1);
      $x2=substr($ris,      strpos($ris,      '=')+strpos($ris,      '?')+2, strlen($ris)-strpos($ris, '?')-1);
      echo "<br />" . $x1 . " " . $x2;
}
```

Facciamo un altro esempio. Si vuole intercettare la correttezza dei dati, `nome, cognome, email e telefono`, inseriti da un utente nei rispettivi campi di un form. In particolare si vuole:

1. Eliminare gli spazi iniziali e finali (funzione trim).
2. Verificare che non sia vuota (funzione empty).

Se scriviamo quindi il seguente codice dovremmo ripeterlo per tutte le variabili che riceveremo dal form:

```
<?php
      // verifichiamo che la variabile esista
      $campo = trim($_campo);
      if(empty($campo)){
          echo 'Il campo è obbligatorio';
      }
?>
```

Conviene allora definire la funzione "controllo" a cui passo un parametro che rappresenta la variabile associata al campo di cui voglio verificarne la compilazione; il risultato della funzione deve essere FALSE se la variabile risulta essere vuota, altrimenti TRUE.
Questo è il codice:

```
<?php
      function controllo($variabile){
            // eliminiamo gli spazi alla variabile ricevuta come parametro

            $variabile = trim($variabile);

            // se  è vuota il risultato sarà FALSE
            if(empty($variabile)){
                  return FALSE;
            }
            return TRUE;
      }

      $campo = controllo($_POST['nome']);
      $campo2 = controllo($_POST['cognome']);
```

```
    $campo3 = controllo($_POST['email']);
    $campo4 = controllo($_POST['telefono']);

    /*se uno dei campi è FALSE*/
    if (!$campo or !campo2 or !campo3 or !campo4) {
       echo "controllo non superato";
    }
    else{
       echo "controllo superato";
    }
?>
```

ALCUNE FUNZIONI NATIVE

Non occorre certamente conoscerle tutte le funzioni native, sono migliaia, ma basta impararne alcune quelle più utili e le altre utilizzarle a seconda delle necessità ricercandole nella rete o dal sito ufficiale del progetto PHP (http://php.net/).

Le funzioni native si possono classificare in base alla struttura dati da gestire:

- Funzioni per gestire le variabili;
- Funzioni per gestire le stringhe;
- Funzioni per gestire i numeri;
- Funzioni per gestire gli array;
- Funzioni per gestire le date;
- Funzioni per gestire i file;
- Funzioni per gestire le cartelle;
- Upload di un file: la variabile $_FILE e funzioni;
- Funzione mail: inviare una mail tramite PHP.

http://www.miniscript.it/articoli/34/le_funzioni_per_gestire_le_variabili.html

http://www.html.it/pag/16690/le-funzioni-in-php-gestire-le-stringhe/

I numeri con php: casuali, arrotondamento e formattazione

La gestione e manipolazione dei dati numerici è un'operazione molto diffusa e che può essere gestita attraverso l'utilizzo di alcune funzioni. In altra sede si è già parlato della sintassi che devono assumere la variabili numeriche e delle funzioni per il controllo dei dati numerici. In questa sede vedremo come generare numeri casuali, come eseguire arrotondamenti e come formattare i numeri.

NUMERI CASUALI

Per generale numeri casuali con php la funzione da impiegare è rand(): essa accetta due parametri opzionali, il valore minimo e il valore massimo.
Di default il valore minimo è 0 mentre quello massimo è conoscibile tramite la funzione getrandmax().

```
1. <?php
2. echo rand(1,10); // un numero casuale compreso fra 1 e 10
3. ?>
```

Nel caso in cui si desideri un numero casuale con un determinato numero di cifre:

```
1. <?php
2. function random_number($cifre)
3.     {
4.     $max_value = bcpow(10, $cifre)-1;
5.     $min_value = bcpow(10, $cifre-1);
6.     //echo $min_value. ' - ' .$max_value . '<br />';
7.     return rand($min_value, $max_value);
8.     }
9.
10. echo random_number(3); // numero casuale di 3 cifre
11. ?>
```

Bcpow(base, exp) → elevamento a potenza

ARROTONDAMENTI

Per eseguire gli arrotondamenti di numeri decimali php mette a disposizione tre funzioni:

- **ceil()**: arrotonda per eccesso ad un numero intero;
- **floor()**: arrotonda per difetto ad un numero intero;
- **round()**: essa a differenza delle precedenti esegue un arrotondamento che avviene per difetto o per eccesso a seconda delle cifre scartate; essa prevede 3 parametri di cui due opzionali; il primo parametro, obbligatorio, è il numero da arrotondare; il secondo, opzionale è il numeri di cifre decimali (dopo la virgola) da conservare, di default pari a zero; il terzo serve per gestire gli arrotondamenti incerti in virtù del fatto che terminano per 5 (vedi esempio), di default PHP_ROUND_HALF_UP.

```
<?php
echo ceil(2.35); // 3
echo floor(2.35); // 2
echo round(2.35); // 2
echo round(2.65); // 3
echo round(2.35, 1); // 2.4
echo round(2.35, 1, PHP_ROUND_HALF_UP); // 2.4
echo round(2.35, 1, PHP_ROUND_HALF_DOWN); // 2.3
echo round(2.35, 1, PHP_ROUND_HALF_EVEN); // 2.4
echo round(2.35, 1, PHP_ROUND_HALF_ODD); // 2.3
?>
```

FORMATTAZIONE DEI NUMERI

A fini di visualizzazione si ha la necessità di far sì che i numeri abbiano una determinata formattazione. Ad esempio *separare le migliaia* con un punto (o con uno spazio) e/o *i decimali* con una virgola. A questo scopo la funzione che è destinata a tale scopo è **number_format()**. Essa accetta uno, due o quattro parametri (non 3). Il primo è il numero che si vuole formattare. Il secondo parametro serve ad impostare il numero di decimali da considerare, di default è 0 (zero) e quindi il numero sarà formattato senza decimali. Il terzo serve per impostare il carattere separatore tra la parte intera e quella decimale, di default è il punto. Il quarto parametro serve per impostare il carattere separatore nelle migliaia, di default è la virgola.

Attenzione: il terzo e il quarto parametro accettano un unico carattere separatore (un unico byte). Giova inoltre sottolineare che la funzione darà come *return una stringa* la quale non è utilizzabile per eseguire operazioni matematiche successive. E' una funzione che, quindi, deve essere impiegata principalmente nella stampa a video. Vediamone di seguito alcuni esempi:

```
1. <?php
2. $numero = 1120.2536;
3. $num1 = number_format($numero); // 1,120
4. $num2 = number_format($numero, 2); // 1,120.25
5. $num3 = number_format($numero, 2, ',', ' '); // 1 120,25
6. $num4 = number_format($numero, 2, ',', '.'); // 1.120,25
7. ?>
```

Formattare un numero … continua

Nell'elaborazione di tabulati si ha spesso necessità di formattare correttamente i numeri. I problemi che spesso si pongono sono i seguenti:

- troncare i decimali
- usare il divisore italiano virgola per separare i decimali
- usare il divisore italiano punto per dividere le migliaia
- aggiungere degli zeri in testa (fillare a zero a sinistra)

Per eseguire tutte queste operazioni ci vengono in aiuto due funzioni php, la number_format e la sprintf.

Troncare i decimali

Per troncare i decimali é sufficiente passare come secondo parametro della funzione number_format il numero di decimali a cui troncare, così questo esempio tronca al secondo decimale, restituendo 1,234.57, in quanto effettua anche l'arrotondamento:

```
$valore = number_format(1234.5678, 2);
```

Nel caso si voglia troncare senza effettuare l'arrotondamento occorre usare oportunamente la funzione floor(), moltiplicando (ad esempio per 100 se si vogliono due decimali) e dividendo per lo stesso valore, ad esempio:

```
$valore = floor((1234.5678) * 100) * .01;
```

E' chiaro che a sua volta il risultato può essere passato alla number_format().

Usare il divisore italiano virgola per separare i decimali e il divisore punto per le migliaia

Per Usare il divisore italiano virgola per separare i decimali e il divisore punto per le migliaia é sufficiente passare come terzo parametro della funzione number_format il carattere separatore dei decimali (in italiano la virgola) e come quarto parametro il separatore delle migliaia (in italiano il punto). Così questo esempio tronca al secondo decimale e formatta in italiano, restituendo 1.234,57:

```
$valore = number_format(1234.5678, 2, ',', '.');
```

Chiaramente se non si vuole che venga presentato il divisore delle migliaia é sufficiente togliere il carattere tra gli apici del quarto parametro:

```
$valore = number_format(1234.5678, 2, ',', '');
```

Aggiungere zeri in testa ad un numero (fillare a zero a sinistra)

Per aggiungere degli zeri in testa ad un numero e quindi fillarlo a zero a sinistra possiamo utilizzare la funzione sprintf. Se, ad esempio, vogliamo avere numeri tutti di 6 cifre fillati con zeri in testa, possiamo fare:

```
$valore = sprintf("%06d", $numeroDaFillare);
```

Per imporre un numero specifico di decimali anche se questi sono zeri, ad esempio invece di scrivere 2,4 si vuole 2,40 occorre … number_format accetta tre parametri: il numero o la variabile in cui è contenuto, un numero che identifica quante porzioni decimali vanno mostrate (nell'esempio nessuna - il valore 2 avrebbe restituito 10.000,00) e l'espressione ',','.' che indica l'utilizzo del carattere punto (.) come separatore delle cifre.