# HTML e PHP

**Appunti di HTML**
(tratto da: Guida HTML di Massimiliano Valente, Manuale Cremisi,
Guida HTML di WebLink di Andrea Bianchi - http://www.web-link.it)

**HTML** – ***Hyper Text Markup Language*** - è il "linguaggio" usato per creare documenti World Wide Web. In effetti non si tratta di un vero e proprio linguaggio di programmazione, quanto di un "sistema di contrassegno" - linguaggio di markup"- i cui comandi sono dei TAG di formattazione del documento (che consentono anche di inserire informazioni multimediali quali immagini, filmati, suoni, musica, etc.) interpretati ed eseguiti dai vari browser (Netscape, Explorer, Opera, etc.).
Un documento HTML può essere scritto utilizzando un normale editore di testi, ad esempio Blocco Note di Windows, o uno dei tanti editori HTML in circolazione (Front Page, …).

Tutti i documenti HTML, composti da una "intestazione" e da un "corpo", hanno la seguente struttura:

```
<HTML>
    <HEAD>
        <TITLE> titolo </TITLE>
    </HEAD>
    <BODY>
        corpo del documento
    </BODY>
</HTML>
```

Tutti gli elementi del documento debbono stare tra i tag
<HTML> e </HTML> (questo per segnalare ai browser che si usa html e non, ad esempio, xml).
L'**intestazione** del documento è delimitata dai tag **<HEAD>** e **</HEAD>** mentre il corpo del documento è delimitato da **<BODY>** e **</BODY>**.

## INTESTAZIONE

L'intestazione del documento, delimitata da **<HEAD> e </HEAD>**, può contenere diverse cose:

- il Titolo, delimitato da **<TITLE> </TITLE>**, che comparirà nella finestra di visualizzazione dei browser e nei "preferiti" (alcuni motori di ricerca, quali Altavista, utilizzano anche i termini del titolo per l'indicizzazione delle pagine);
- i Termini Chiave per i motori di ricerca, definiti con tag **<META>** con la seguente struttura:

  **<META Name="*nome*" Content="*contenuto*">**

  *nome* = : keywords / author / description / generator / robots endif;;

- il tipo di html utilizzato;
- i link base di riferimento.

## CORPO

Il corpo del documento, delimitato da **<BODY> e </BODY>**, contiene il documento vero e proprio.

Si può impostare il colore di sfondo con **BGCOLOR="*colore*"** (*colore* è il termine inglese del colore) e anche un'immagine sfondo con **BACKGROUND="*file immagine*"** (*file immagine* è un file che si trova sulla stessa directory del documento).

Esempio:

**<BODY BGCOLOR="red" BACKGROUND="foto.gif">**

- Il colore del testo, di default nero, si imposta con **TEXT**="*colore*".
- Il colore dei link, di default blue, si imposta con **LINK**="*colore*".
- Il colore dei link cliccati si imposta con **ALINK**="*colore*".
- Il colore dei link visitati si imposta con **VLINK**="*colore*".

## FORMATTAZIONE

In generale, i tag servono non solo a marcare elementi strutturali del documento, ma anche a dare una formattazione ai vari elementi che compaiono nel documento:

- **<I> </I>** *corsivo*
- **<B> </B> grassetto**
- **<U> </U>** sottolineato
- **<FONT> </FONT>** imposta il font. Attributi:Face="*font*", Size=*grandezza*, Color=*colore*.

I tag che marcano i diversi elementi strutturali del documento ed i tag di formattazione sono classificati come "**tag fisici**". HTML prevede anche dei "**tag logici**" che contrassegnano determinate strutture e che non sono interpretati dai browser per la formattazione. Alcuni di questi tag logici sono:

- <ADDRESS> ( e </ADDRESS>) contrassegna un indirizzo
- <BLOCKQUOTE> contrassegna citazioni lunghe più di due o tre righe
- <CITE> contrassegna la fonte della citazione
- <CODE> contrassegna righe di codice di programmazione
- <VAR>, legato a code, identifica le variabili delle istruzioni
- <STRONG> enfatizza il testo

Al testo si posso anche aggiungere effetti diversi da corsivo, grassetto etc. L'effetto evidenziatore, ad esempio, si ottiene aggiungendo uno **sfondo al testo:**

**<span style**="BACKGROUND-COLOR:#FFFF00">Questo è un testo di prova****

## PARAGRAFI

Il tag <P> </P> definisce e delimita un paragrafo di testo che, con ALIGN=*left*/*center*/*right*, può essere allineato a sinistra, centrato, a destra. Per "andare a capo" si utlizza il tag <BR>.
Testo ed immagini possono essere allineati anche con **<DIV> </DIV>** e la clausola ALIGN.

Con il tag <HR> si possono inserire linee orizzontali.

## ELENCHI Puntati e Numerati

Gli **elenchi numerati** sono delimitati da **<OL> </OL>** e da tanti tag <**LI**> quante sono le voci dell'elenco.
Nel tag OL, con la clausola **TYPE** si specifica il **tipo di elenco**:

- "A" (lettere maiuscole),
- "a" (lettere minuscole),
- "I" (numeri romani maiuscoli),
- "i" (numeri romani minuscoli).

Nel caso in cui si volesse **partire da un qualsiasi numero diverso da "1"**, o da una **lettera diversa da "a"**, sarà sufficiente fare uso dell'attributo **START** impostando il numero da cui iniziare l'incremento:

```
<OL START="3">
   <LI>Rosso
   <LI>Bianco
   <LI>Verde
   <LI>Nero
```

Questo è il risultato:

3. Rosso
4. Bianco
5. Verde

**</OL>**

6. Nero

Vediamo **un esempio** anche nel caso in cui si adoperino le lettere. Dovendo iniziare dalla lettera "C" ed essendo questa la terza lettera dell'alfabeto, si dovrà inserire il numero "3":

```
<OL TYPE="a" START="3">
    <LI>Rosso
    <LI>Bianco
    <LI>Verde
    <LI>Nero
</OL>
```

Questo è il risultato:

c) Rosso
d) Bianco
e) Verde
f) Nero

Per gli **elenchi puntati** si utilizzano i tag **<UL> e </UL>,** con la clausola **TYPE** si specifica il tipo di punto:

- "disc" (punti pieni),
- "circle" (punti vuoti),
- "square" (quadrati)

Esempio

```
<UL TYPE="circle>
    <LI> Informatica
    <LI> Matematica
    <LI> Economia Aziendale
</UL>
```

Questo è il risultato:

- Informatica
- Matematica
- Economia Aziendale

## IMMAGINI

Le immagini (.gif, .bmp, .jpg) si inseriscono utilizzando il tag <IMG>. Non si tratta di un vero e proprio "inserimento", ma di un "collegamento" al file grafico, locale o sul web. Questo vale anche per altri tipi di informazioni multimediali (suoni, musica, filmati, applet, etc.).

**<IMG SRC="foto.jpg">** dove SRC specifica il nome, comprensivo di percorso, del file.

Nel tag IMG si possono utilizzare diverse opzioni di formattazione, tra le quali:

| | |
|---|---|
| **ALT="commento"** | per inserire commenti |
| **WIDTH=*dimensione*** | dimensione orizzontale |
| **HEIGHT=*dimensione*** | dimensione verticale |
| **BORDER=*tipo*** | tipo bordo (0-nessuno, 1-bordo, 2-bordo doppio) |
| **HSPACE=*distanza*** | distanza orizzontale dagli altri oggetti |
| **VSPACE=*distanza*** | distanza verticale dagli altri oggetti |
| **ALIGN=*allineamento*** | allineamento rispetto al testo (top, middle, bottom, left, right) |

## AUDIO

Per inserire file audio (.wav, mp2, .mpeg, mdi) si utilizza Bgsound:

**<BGSOUND SRC="*file audio*">**

Con il tag **EMBED**, si può rendere automatica l'apertura del file audio.

## FILMATI

Per inserire filmati (.avi) si utilizza Img:

**<IMG DYNSCR="*file video*">**

## LINK

I link sono gli elementi che consentono di fare dei documenti HTML documenti IPERTESTUALI, permettendo di inserire "legami" – HyperLink - con punti diversi all'interno dello stesso documento o, soprattutto, con altri documenti HTML locali o nel web. Questo insieme di legami tra documenti localizzati in computer sparsi per il mondo costituisce la "ragnatela mondiale", il World Wide Web.
Il tag per linkare un altro documento è:

**<A HREF="*url o pagina*">descrizione</A>** (A=Anchor, HREF=HyperText Reference)

| | |
|---|---|
| <A HREF="http://www.tin.it">Tin</A> | crea un link al sito www.tin.it |
| <A HREF="Storia.Htm">Storia</A> | crea un link al file html Storia.Htm |
| <A HREF="#capdue">Cap.2</A> | crea un link al punto capdue interno al documento |
| <A HRAF="storia.htm#grecia">grecia</a> | crea un link al punto Grecia del documento Storia Htm |
| <A HREF="mailto:ugo@tin.it">Scrivi</a> | attiva il programma di posta elettronica pronto ad inviare una mail all'indirizzo ugo@tin.it |

## TABELLE

Le tabelle, oltre che per contenere dati, vengono utilizzate soprattutto per posizionare elementi nel documento.
Per definire tabelle si usano i tag **<TABLE> e </TABLE>** (attributi: Border, Width, Align)
Con i **<CAPTION>titolo</CAPTION>** si può assegnare un titolo alla tabella (attributi: Top, Bottom)
Con i tag <TH> e <TH> si possono assegnare intestazioni alle colonne
Con i tag <TD> e </TD> si inseriscono i dati nella tabella
(attributi di TH e di TD: Align, Valign, Width, Height, Nowrap, Colspan, Rowspan)
Con i tag **<TR> e </TR>** si specificano le righe della tabella

## FORM (o MODULI)

La maggior parte degli elementi di Html permette al visitatore del sito la visualizzazione del suo contenuto, ma non permette di interagire con esso.
Con i **MODULI**, invece, l'utente può interagire con il sito (inviando un commento, facendo richieste, firmando guestbook, rispondendo a domande, etc.).
**Questa interazione è però possibile solo perché residenti nel server ci sono dei programmi, ad esempio, CGI (Common Gateway Interface), PHP (Hipertext Preprocessor) etc. che la realizzano interpretando ed eseguendo i FORM.**

I tag per la creazione del form sono **<FORM>** e **</FORM>** che delimitano ciascun modulo. Non è possibile racchiudere un form dentro un altro form.

**<FORM METHOD="Get" ACTION="http://www.sito.it/cgi-bin/scriptxyz.cgi">**
**<FORM METHOD="Post" ACTION="inserisci.php">**

A questi tag è sempre associato:
- l'attributo **ACTION** che determina l'utilizzo dei dati d'input trasmessi tramite il modulo,
- uno fra i due metodi **GET** e **POST.**

Vediamo ora come creare un form e richiederne l'esecuzione.

1. La creazione di un form avviene in due fasi:
   1. impostazione dei tag per la creazione del modulo, dei campi e del tasto di spedizione
   2. creazione di uno script CGI sul server o richiamo di uno script PHP già impostato.
2. Quando l'utente inoltra un modulo (ad es., attivando un **pulsante d'invio**), l'interprete lo elabora seguendo quattro fasi:
   1. **identificazione dei controlli a buon esito**.

2. **costruzione di un insieme di dati del modulo**: un insieme di dati del modulo è una sequenza di coppie **nome-di-controllo/valore-attuale** costituita da controlli a buon esito.
3. **codifica dell'insieme di dati del modulo**: l'insieme di dati del modulo viene codificato in accordo con il tipo di contenuto specificato dall'attributo **enctype** dell'elemento FORM.
4. **invio dell'insieme di dati del modulo così codificato**: al termine i dati codificati vengono inviati al programma di elaborazione indicato dall'attributo action usando il protocollo specificato dall'attributo **method**.
   - se il **method** è "**get**" e **l'action** è un **URL** dell'HTTP, l'interprete prende il valore di **action**, gli aggiunge un **?** e poi ci aggiunge l'**insieme di dati del modulo,** codificato usando il tipo di contenuto "application/x-www-form-urlencoded". L'interprete poi si sposta verso tale URI. In tale situazione i dati del modulo sono limitati ai codici ASCII;
   - se il **method** è "**post**" e **l'action** è un **URL** dell'HTTP, l'interprete conduce un'operazione di **transazione** "post" in HTTP usando il valore dell'attributo action ed un messaggio creato in accordo con il tipo di contenuto indicato dall'attributo **enctype**.

Gli interpreti dovrebbero presentare il responso alle operazioni "get" e "post" dell'HTTP.

**Nel modulo è possibile creare alcuni elementi di gestione dati:**

Il tag base per la definizione degli elementi di un form è **<INPUT>** che può assumere 7 diversi valori.

**Type="Text"** <INPUT Type="Text" Name="nome" Maxlenght="80" Value="tuo nome">
Crea tipici campi di testo – textbox – nei quali richiedere informazioni

**Type="Password"** <INPUT Type="Password" Name="nome" Maxlenght="10">
Crea tipici campi di testo – textbox – mascherati con asterischi

**Type="Checkbox"** <INPUT Type="Checkbox" Name="allegati" Value="Yes" checked>
Crea caselle - checkbox - per informazioni di tipo si/no (checked controlla lo
stato della casella al caricamento della pagina)

**Type="Radio"** <INPUT Type="Radio" Name="giudizio" Value="non sufficiente">
<INPUT Type="Radio" Name="giudizio" Value="sufficiente">
<INPUT Type="Radio" Name="giudizio" Value="buono">
Crea cerchietti – optionbutton – per scelte multiple. Se hanno tutti lo stesso
nome, selezionandone uno si deselezionano automaticamente gli altri.

**Type="Image"** <INPUT Type="Image" Src="foto.gif">
Funziona come Submit, ma usa una immagine al posto del pulsante.

**Type="Hidden"** <INPUT Type="Hidden", Name="MAILFORM_SUBJECT" Value="titolo del form">
determina il titolo del messaggio, **invisibile all'utente**, che sarà ricevuto via e-mail e che conterrà il contenuto del form
<INPUT Type="Hidden", Name="MAILFORM_URL"
Value="http://www.sito.it">
una volta compilato e spedito il form, da in risposta una pagina html successiva all'interno della quale è possibile inserire dei commenti **invisibile all'utente**.

**Type="Submit"** <INPUT Type="Submit" Value="Invia">
Crea pulsante – commandbutton - per l'invio del form.

**Type="Reset"** <INPUT Type="Reset" Value="Azzera">

Crea pulsante – commandbutton - per l'azzeramento dei dati inseriti.

Il tag <TEXTAREA> è usato per definire campi che prevedono molto testo, su più righe:
<TESTAREA Cols=40 Rows=5 Wrap="phisical" Name="commento"></TEXTAREA>
Con phisical si inserisce l' "a capo" automatico.

Il tag <SELECT> permette di definire elenchi a discesa – combobox – con varie possibilità di scelta:

```
<SELECT Size=1 Name="Colori">
<OPTION>Rosso
<OPTION>Bianco
<OPTION>Verde
</SELECT>
```

**Appunti di PHP**
(tratto da: http://www.webmasterpoint.org/php/)

## Introduzione

Il PHP è un linguaggio implementato lato server (server-side HTML-embedded scripting language). Il suo funzionamento è molto semplice ed efficace.
L'engine del PHP dà come risposta delle richieste al Web Server (nel nostro caso Apache) pagine HTML completamente formattate, rendendo il codice PHP perfettamente trasparente all'utente finale.

Visualizzando il codice delle pagine si vedrà solo HTML puro e di fatto il sorgente della pagina PHP può essere modificato solo dal Web Master.

## Com'è nato PHP?

Il PHP nasce a metà del 1994 dalle mani di Rasmus Lerdorf. Egli sviluppo una prima versione mai ufficializzata che utilizzò sulle sue pagine. Ma il successo non tardò, e già nella prima parte del 1995 usci la prima versione del **P**ersonal **H**ome **P**age. Da lì a poco l'engine, anche grazie alla sua filosofia free, si è affermato.

## Come inserire codice php

Come sappiamo il codice PHP è interpretato dall'engine (**parser**) che si presenta come un modulo del Web Server, nel nostro caso Apache. Ma come fa l'engine a capire cose deve elaborare?
La prima cosa è sicuramente l'impostazione dell'estensione del file php. Infatti quando al Web Server viene chiesta una pagina con estensione .php egli gira la pagina direttamente all'engine, che ha poi il compito di capire cosa interpretare e di cosa lasciare così com'è e di restituirci la pagina. L'engine capisce cosa processare tramite un tag particolare. Questo particolare "codice" è ritrovabile in 4 modi differenti. I due metodi più usati sono:

```
<?php
        ...codice php...
?>
o

 <?
        ...codice PHP...
?>
```

Tutto quello che si trova all'interno dei tag verrà elaborato dal parser.

Esempio:

```
<html>
 <head>
  <title>...</title>
 </head>

 <body>
  <!-- codice HTML non interpretato -->

   <?
     / * codice PHP interpretato * /
   ?>

  <!-- codice HTML non interpretato -->
 </body>
```

```
</html>
```

## Le prime funzioni PHP

Le prime funzioni che vogliamo subito provare sono quelle di output o stampa a video. Vediamo subito un esempio:

```
< ?PHP
        print 'Ciao gente!';
?>

< ?PHP
        echo 'Ciao gente!';
?>
```

All'esecuzione di ciascuno di questi script, verrà visualizzata la scritta "Ciao gente !".

Possiamo anche usare **caratteri speciali** \n o la direttamente la **tag html <br>** nel seguente modo:

```
< ?PHP
        echo "Stampo tutto su<br> più righe";
?>
```

**Le virgolette doppie**, che servono ogni qualvolta ci sono caratteri speciali da interpretare o variabili stringa da interpolare.
Quindi, per stampare **stringhe statiche**, meglio usare gli **apici singoli**, che hanno un'esecuzione leggermente più veloce, ma che può risultare sensibile nei casi di script molto complessi e lunghi.

## Le variabili scalari e vettoriali

Uno degli aspetti base di ogni linguaggio di programmazione sta nelle variabili. **Le variabili del PHP sono molto flessibili e vengono riconosciute automaticamente**. La variabile PHP viene definita nel suo contenuto dall'engine. Le variabili PHP possono contenere di tutto.

Come per gli altri linguaggi di programmazione, anche PHP utilizza **variabili scalari**, comunemente dette **variabili**, e **variabili vettoriali**, comunemente dette **vettori** (array). I vettori possono essere monodimensionali (vettori veri e propri) o multidimensionali (comunemente detti matrici).

**L'inizializzazione di una variabile può essere fatta semplicemente assegnandole un valore**, cioè usando l'**operatore =**. A volte però può essere utile o necessario assegnare anche il tipo alla variabile, questa è la sintassi:

```
$variabile = (int) 1;
```

Al posto di **int** è possibile sostituire altri tipi di variabili.

L'inizializzazione di un array (vettore) è fatta assegnando un valore, ad esempio:

```
$nomi[ ] = "Davide"    // Questo valore è identificato come array $nomi[0]
$nomi[ ] = "Fabio"     // Questo valore è identificato come array $nomi[1]
.... così via!
```

In questo modo quando viene inserito un nuovo valore nell'array, questo diventa l'ultimo. **E' molto importante ricordare che la posizione degli elementi di un array parte da 0 e non da 1**.

Esistono inoltre le così dette **variabili di sistema** fra cui vale la pena di mettere in evidenza:

- **$HTTP_HOST**: la variabile che mostra il **nome del dominio** del sito su cui lo script viene eseguito
- **$PHP_SELF:** la variabile mostra **nome e path** dello script in esecuzione

## Gli operatori

Con le variabili è possibile effettuare le "classiche" **operazioni aritmetiche:**

| Simbolo | Cosa fa | Esempio con $x=10 e $y=4 | Risultato |
|---|---|---|---|
| + | Somma i due operandi | $x+$y | 14 |
| - | Sottrazione | $x-$y | 6 |
| * | Moltiplica i due operandi | $x*$y | 40 |
| / | Divisione | $x/$y | 2.5 |
| % | Resto divisione intera | $x%$y | 2 |

L'operatore **= consente di assegnare un valore ad una variabile**. In particolare:

- **+=**
- **/=,**
- **-=,**
- ***=,**
- **%=**

Ad esempio: **$x+=3;**
che equivale a scrivere: **$x=$x+3;**

Esiste anche l'operatore di **concatenazione di stringhe**:

```
$messaggio = 'Il cliente Mario Rossi ha comprato i seguenti articoli:<br>';
$messaggio .= 'Tappeto persiano codice T135';
```

che equivale a scrivere:

```
$messaggio = 'Il cliente Mario Rossi ha comprato i seguenti articoli:<br>' . 'Tappeto persiano codice T135';
```

Ci sono poi gli **operatori incrementali e decrementali**, molto utili per alleggerire le istruzioni:

```
$x=10;
$x++; //$x contiene 11
$x++ equivale a $x=$x+1

$x--;  //$x contiene 10
$x-- equivale a $x=$x-1
```

Bisogna precisare però che in questo caso, **la variabile viene memorizzata** (in un eventuale variabile a sinistra) **e poi incrementata**.
Infatti,

```
$x=10;
$x++;      //$x contiene 11
$x++ equivale a $x=$x+1

$x--;      //$x contiene 10
$x-- equivale a $x=$x-1
```

```
$y=$x++;
```

con questa istruzione, la variabile $y conterrà il valore di $x prima dell'incremento.

```
$y=++$x;
```

al contrario, con questa istruzione la variabile $y conterrà il valore di $x dopo l'incremento.

## Il controllo IF

L'istruzione IF... THEN... ELSE... è una istruzione che mette il computer nelle condizioni di decidere che tipo di operazione compiere in base a delle condizioni da verificare. E' ovvio che utilizza un sistema binario true/false per eseguire le sue scelte.
L'istruzione IF può essere utilizzata in vari modi.
I primo metodo consiste nel fare eseguire una istruzione solo se si verifica una condizione particolare e poi ritornare nella linearità del programma, che viene invece mantenuta se l'IF da come risposta un false.

Questo tipo di funzionamento si ottiene inserendo all'interno del listato questa funzione:

```
if (espressione):
        istruzioni
endif;
```

Facciamo un esempio stupido:

```
if ($a = 4):
  $b = $a/2;
  $a = $b;
endif;
```

Con questo semplice listato abbiamo inserito una condizione. La variabile $a viene divisa per due nel caso in cui il suo valore sia 4 e non viene invece modificata nel caso in cui il suo valore sia diverso da 4.

Un altro metodo di usare l'IF è quello classico di effettuare una data operazione x se if = true e di eseguirne una altenativa se if = false.

Questo è il listato base di questa funzione:

```
if (espressione):
        istruzioni x
else:
        istruzioni alternative non-x
endif;
```

Oltre a questi ci sono altri metodi che implementano l'utilizzo dell' **elseif**.

## Il ciclo While - While/Do

Il ciclo **WHILE** serve per eseguire un blocco di istruzioni al verificarsi di una determinata condizione. Ecco qui il codice base:

```
while (espressione):
        istruzione
endwhile;
```

...ed un piccolo classico esempio:

```
<?
    $i = 0;
    while ($i <= 10):
        echo $i;
        echo "<br>";
        i++;
```

```
    endwhile;
?>
```

...che visualizza i primi dieci numeri naturali.

Esiste anche un altro metodo con cui è possibile realizzare l'istruzione While: è il sistema **WHILE - DO**.

La sua sintassi è leggermente differente:

```
do {
        espressione
}
while (istruzione)
```

Ecco qui il precedente esempio riscritto con la nuova sintassi:

```
<?
    $i =0;
    do
    {
            echo $i;
            echo "<br>";
            i++;
            }
    while ($i <= 10);
?>
```

La differenza dei due cicli sta nel fatto che:

- nel primo ciclo l'istruzione non viene effettuata neanche una volta se al primo "giro" si riceve un valore "false",
- nel secondo caso le istruzioni vengono comunque eseguite almeno una volta.

Questa differenza, da non trascurare, si trova nella posizione dell'istruzione di controllo. Infatti nel primo caso il controllo avviene a monte mentre nel secondo caso il controllo avviene a valle.

## Il ciclo For

Vediamo ora il **ciclo FOR**. Questo ciclo è più flessibile e potente dei costrutti di controllo del PHP. Il costrutto logico della scrittura è lo stesso del costrutto WHILE perché il controllo dell'espressione è eseguito a monte. Ecco qui la sintassi del ciclo:

```
for (expr1; expr2; expr3):
        istruzione
endfor;
```

in cui:

- expr1 è una istruzione che viene valutata prima di eseguire il ciclo,
- expr2 è l'espressione che viene valutata per decidere se il ciclo deve continuare o meno,
- expr3 è l'operazione che viene eseguita al termine di ogni ciclo.

Facciamo un esempio pratico. Eseguiamo con il ciclo FOR lo stesso algoritmo che abbiamo creato con WHILE. Ecco il listato:

```
<?
        for ($i = 0 ; $i <= 10 ; $i++):
                echo  $i ;
                echo  "<br>";
```

```
        endfor;
?>
```