PHP

Introducere în limbajul PHP

PHP este abrevierea de la Personal Home Page şi reprezintă un limbaj creat în 1994 de către Rasmus Lerdorf pentru pagina personală. Odată cu creşterea numărului de utilizatori, PHP şi-a schimbat semnificația în Hypertext Preprocessor.

PHP este un limbaj de redactare a scripturilor înglobat în HTML ce permite dezvoltatorilor web să realizeze pagini web dinamice (http://www.php.net). Afirmația că PHP este înglobat în HTML se referă la faptul că poate fi scris în interiorul codului HTML al paginii web. PHP este creat pentru a efectua o anumită operație numai după producerea unui eveniment – de exemplu când un utilizator trimite un formular sau se deplasează către un URL.

Printre avantajele utilizării PHP pot fi enunțate: este Open Source, poate fi uşor de învățat datorită sintaxei care se aseamănă cu C, Java şi Perl, operațiile care se efectuează sunt pe parte de server, poate interacționa cu baze de date şi fişiere, poate manipula mesaje de poştă electronică, poate fi folosit pe aproape orice sistem de operare şi permite transpunerea fişierelor dintr-o platformă în alta cu foarte mici modificări. Ca dezavantaje putem enunța: nu poate fi folosit pentru caracteristicile pe parte de client aşa cum este folosit JavaScript, poate fi folosit numai de pe un server care poate citi codul PHP.

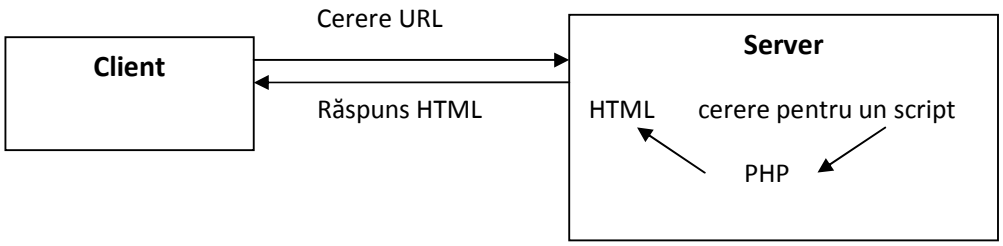


Figura 3.5.1. Proces client – server PHP

În figura 4.5.1. este prezentată cererea unui client către un server unde este funcțional PHP care este urmată de executarea operațiilor pe parte de server şi transmiterea informațiilor corespunzătoare către browser în format HTML.

De ce avem nevoie?

Pentru a putea folosi PHP, avem nevoie de acces la un server unde acest limbaj este funcțional. Exemple de aplicații de server Web sunt: Apache pentru Linux şi MacOS X sau IIS pentru Windows. Aceste aplicații pot fi instalate pe computerul propriu sau pot fi închiriate şi folosite serviciile PHP oferite de diverse companii.

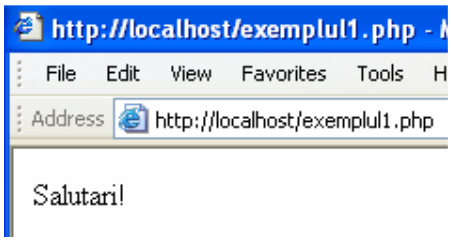
Noțiuni elementare de sintaxă PHP

În exemplele următoare vom utiliza un server instalat pe computerul personal, la care avem acces prin http://localhost/numele_fisierului.php.

Un fişier PHP poate avea extensiile: ".php", ".php3" sau ".phtml". Codul unui fişier PHP este încadrat de comenzile de început şi sfârşit: <?php şi ?>, ca în exemplul de mai jos.

Exemplu de pagină Web scrisă cu PHP.

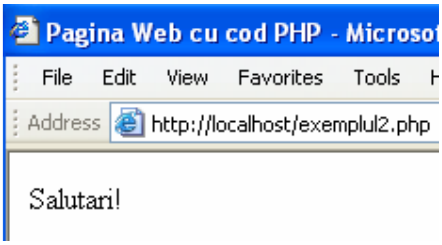
```
<?php
print"Salutari!";
?>
```



Totodată, acelaşi cod poate fi inserat în interiorul unui cod HTML, după cum veți vedea în exemplul de mai jos.

Exemplu de pagină Web în care se inserează cod PHP.

```
<html>
<head>
<title>Pagina Web cu cod PHP</title> </head>
<body>
        <?php
        print"Salutari!";
        ?>
</body></html>
```

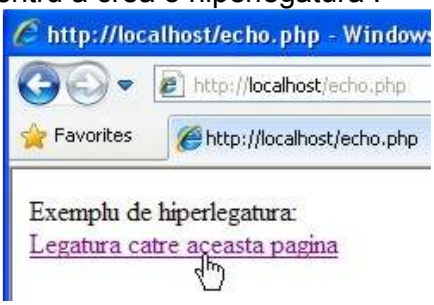


OBS *PHP rulează pe diverse platforme, cum ar fi: Windows, Linux, Unix.*

Tot pentru a afişa un mesaj într-o pagină Web, putem utiliza funcția *echo()*.

Exemplu de pagină PHP în care se introduce cod HTML pentru a crea o hiperlegatură .

```
<?php
   echo 'Exemplu de hiperlegatura:  </br>
   <a href="echo.php">Legatura catre aceasta
      pagina</a>';
?>
```



Comentarii

Pentru a introduce un comentariu într-un cod PHP putem folosi

// comentariu pentru o singură linie de cod

sau

/* exemplu
   de comentariu pe mai multe rânduri */

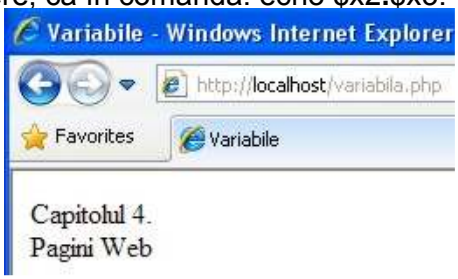
Variabile

O variabilă îşi poate modifica valoarea. Toate variabilele din PHP încep cu caracterul $ şi se termină cu ;. Numele unei variabile trebuie să înceapă cu o literă sau cu underscore (_), să conțină numai caractere alfanumerice şi underscore (a-z, A-Z, 0-9, _) şi nu trebuie să conțină spații.

Exemplu: $nume_var=' '; sau $nume_var=" ";

Exemplu În exemplul de mai jos am inițializat variabilele $x1, $x2 şi $x3 cu câte un şir de caractere pe care ulterior le-am afişat cu ajutorul comenzii echo(). Caracterul punct (.) permite concatenarea şirurilor de caractere, ca în comanda: echo $x2.$x3.

```
<html>
<head>
<title>Variabile</title></head>
<body>
        <?php
        $x1="Capitolul 4.";
        $x2="Pagini";
        $x3=" Web";
        echo $x1."<br />";
        echo $x2.$x3;
        ?>
</body>
</html>
```



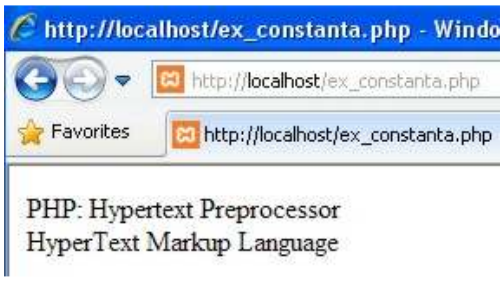
Constante

O constantă poate reține o valoare care nu mai poate fi modificată în cadrul scriptului. Pentru a defini o constantă folosim funcția **define();**

OBS O diferență între scrierea unei variabile şi a unei constante este faptul că o constantă nu are în fața ei semnul $.

Exemplu În exemplul de mai jos am afişat conținutul a două constante definite: PHP şi HTML..

```
<?php
define('PHP', 'PHP: Hypertext
    Preprocessor');
define('HTML', 'HyperText Markup
    Language ');
echo PHP.' <br> '.HTML;
?>
```



Tipuri de variabile

Variabilele pot fi de mai multe tipuri:

- scalare: boolean, integer, float, string
- compuse: array, object
- speciale: resource, null

În PHP au mai fost introduse trei pseudotipuri: mixed, number şi callback.

Tipul unei variabile în PHP nu trebuie neapărat specificat de către programator, pentru că poate fi stabilit în timpul execuției codului respectiv.

Caractere speciale

În codul PHP se pot include şi caractere speciale, precum :

- \n - salt la linie noua
- \r – carriage return
- \t - caracter de tabulare pe orizontala
- \\ - backslash
- \$ - simbolul dolarului
- \" - ghilimele duble

Operatorii PHP pot fi clasificați în operatori aritmetici, de atribuire, de comparație şi logici.
Operatorii aritmetici

Operator	Descriere	exemplu
+	Adunare	x+1
-	scădere	x-1
*	înmulțire	x*3
/	împărțire	x/3
%	rest	10%3
++	incrementare	x++
--	decrementare	x--

Operatori de atribuire

operator	exemplu	Exemplu similar
=	x=3	x=3
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Operatori de comparație

operator	descriere
==	Este egal cu
!=	Nu este egal cu
<>	Nu este egal cu
>	Mai mare decât
<	Mai mic decât
>=	Mai mare sau egal
<=	Mai mic sau egal

Operatori logici

operator	descriere
&&	Şi logic
||	Sau logic
!	Negație logică

Funcții PHP

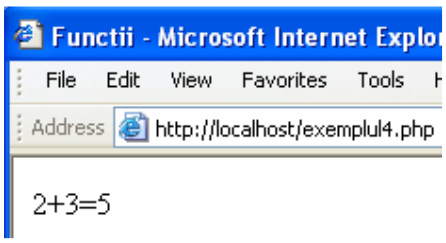
Funcțiile PHP pot fi create de noi sau putem utiliza funcțiile predefinite ale limbajului, cum ar fi: *echo(), eval(), print()* şi multe altele.

Dacă dorim să creăm funcții PHP, trebuie mai întâi să le definim, după care le putem utiliza prin apelarea lor.

Definirea unei funcții constă în cuvântul cheie **function**, urmat de: numele funcției, o listă cu parametri separați prin virgulă şi instrucțiunile scrise între acolade. O funcție poate fi apelată de către o altă funcție sau de către ea însăşi (se autoapelează).

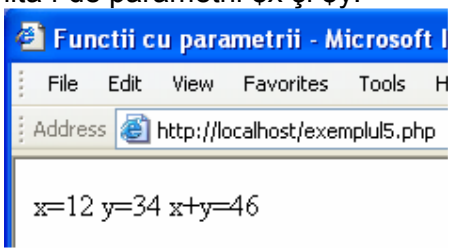
Exemplu În acest exemplu am creat o funcție ce realizează adunarea conținutului a două variabile: $x1 şi $x2. Apelarea funcției se face prin numele ei: *f(),* după definirea acesteia. Tot aici putem observa faptul că trebuie încadrat conținutul funcției între acolade: { }.

```
<html>
<head>
<title>Functii</title></head>
<body>
        <?php
        function f()
        {
              $x1=2;
              $x2=3;
               echo "2+3=";
```



```
            echo $x1+$x2;
        }
        f();
        ?>
</body>
</html>
```

Exemplu În acest exemplu se apelează o funcție definită f de parametrii $x şi $y.




```
<html>
<head>
<title>Functii cu parametrii</title>
</head>
<body>
        <?php
        function f($x,$y)
        {
            echo " x=12\ny=34\nx+y=";
            echo $x+$y;
        }
        $x=12;
        $y=34;
        f($x,$y);
        ?>
</body>
</html>
```

Limbajul PHP este amplu şi nu mi-am propus abordarea lui pe larg aici. Ceea ce am dorit să demonstrez este faptul că se poate utiliza relativ uşor într-o pagină Web, dacă avem la dispoziție un server precum Apache sau IIS.

Funcția $_GET este utilizată pentru a colecta valori într-un formular cu metoda “get”. Informația trimisă dintr-un formular cu metoda GET este vizibilă tuturor, adică va fi afişată în bara de adrese din browser şi nu poate suporta mai mult de 100 caractere. Când utilizăm metoda “get” dintr-un formular HTML, toate numele variabilelor şi valorile vor fi afişate în URL, de aceea nu trebuie utilizat atunci când intenționăm să transmitem parole sau date confidențiale.

Funcția $_POST este utilizată pentru a colecta valori într-un formular cu metoda “post”. Informația trimisă dintr-un formular cu metoda POST este invizibilă celorlalți utilizatori, iar cantitatea de informație transmisă nu este limitată. Un dezavantaj ar fi faptul că paginile în care se utilizează nu pot fi reținute cu bookmark, deoarece variabilele nu sunt afişate în URL.

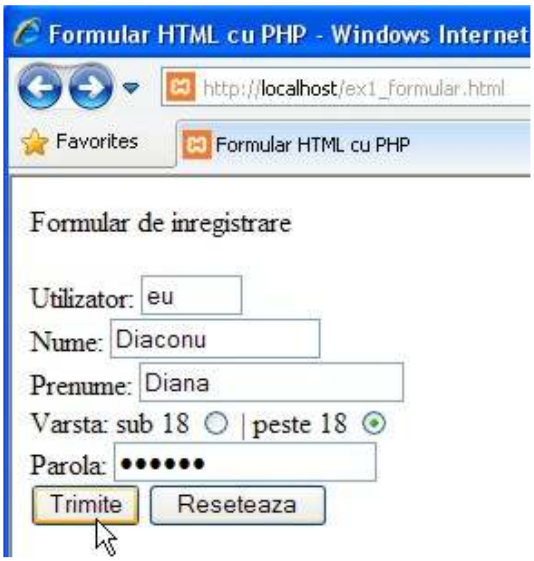
Funcția $_REQUEST poate fi utilizat pentru a colecta date dintr-un formular şi a le trimite cu metodele GET sau POST.

Formulare cu HTML şi PHP

În cadrul capitolului 4.1. HTML am parcurs paşii necesari pentru crearea unui formular. În interiorul etichetei <form> am introdus elementele ***action*** şi ***method*** (care poate fi POST sau GET). Elementul ***action*** conține calea către scriptul PHP care prelucrează datele iar ***method*** conține metoda prin care se vor prelucra datele atunci când este acționat butonul “Apasati”.

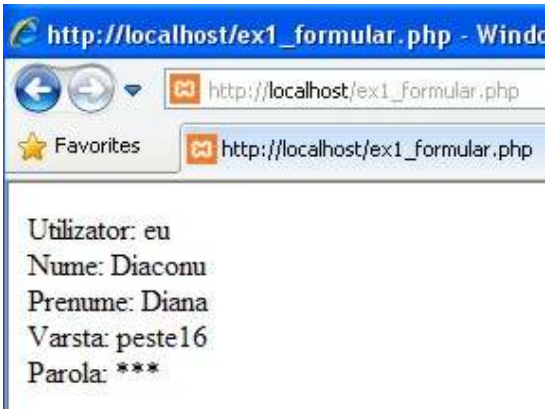
Exemplu În exemplul de mai jos, am creat un formular în HTML care va fi salvat cu numele: ex1_formular.html şi un cod PHP care va fi salvat cu numele ex1_formular.php.

```
<html>
<head><title>Formular HTML cu
    PHP</title></head>
<body>
Formular de inregistrare<br /><br />
<form action="ex1_formular.php"
    method="post">
Utilizator: <input type="text"
    name="utiliz" value="" size=5
    maxLength=5> </br>
Nume: <input type="text" name="nume"
    value="" size=15 maxLength=15> </br>
Prenume: <input type="text"
    name="prenume" value=""> </br>
Varsta: sub 18 <input type="radio"
    name="varsta" value="sub18"> | peste
    18
<input type="radio" name="varsta"
    value="peste18"> </br>
Parola: <input type="password"
    name="parola" value=""> </br>
<input type="submit" name="Apasati"
    value="Trimite">
<input type="reset" name="Reseteaza"
    value="Reseteaza">
</form>
</body>
</html>
```



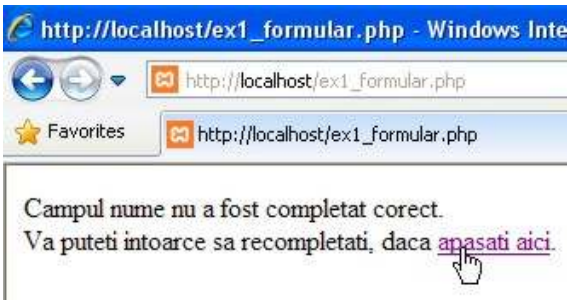
Cod PHP

```
<?php
echo 'Utilizator:
    '.$_POST['utiliz'].'<br>
Nume: '.$_POST['nume'].' <br>
Prenume: '.$_POST['prenume'].' <br>
Varsta: '.$_POST['varsta'].' <br>
Parola: *** <br>';
?>
```



Verificarea datelor introduse în formular se face îmbunătățind codul PHP, aşa cum se observă în exemplul următor:

```
<?php
if(($_POST['nume'] == "") ||
      (is_numeric($_POST['nume'])) ||
      (strlen($_POST['nume']) < 3))
 {echo 'Campul nume nu a fost completat
      corect. <br> Va puteti intoarce sa
      recompletati, daca <a
      href="ex1_formular.html">apasati
      aici</a>.'; }
else
 { echo 'Utilizator:
      '.$_POST['utiliz'].'<br>
Nume: '.$_POST['nume'].' <br>
Prenume: '.$_POST['prenume'].' <br>
Varsta: '.$_POST['varsta'].' <br>
Parola: *** <br>'; }
?>
```



Din ultimul exemplu putem observa că se face verificarea caracterelor introduse în câmpul nume. Similar se pot verifica şi celelalte câmpuri.

Formular pentru verificarea adresei de e-mail

În exemplul de mai jos se găseşte un formular ce permite trimiterea unui mesaj către o adresă de e-mail.

Exemplu de formular. După ce utilizatorul completează câmpurile formularului, se apelează un fişier cu numele: "trimite_f.php" care se află în acelaşi folder cu fişierul ce conține formularul.

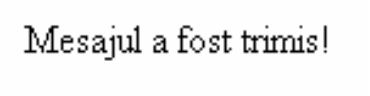
```
<html>
<head><title>Formular</title></head>
<body>
Exemplu de formular<br /><br />
<FORM action="trimite_f.php" method=
    post>
Numele:<br /><INPUT size=40
    name="nume"> <br />
Adresa de e-mail: <br /><INPUT
    size=40 name="email"><br />
Opinie:<br />
  <TEXTAREA name="mesaj" rows=3
    cols=40></TEXTAREA><br /> <br />
<INPUT type="submit"
    value="Trimiteti!"> </FORM>
</body>
</html>
```

Exemplu Mai jos este prezentat fişierul: "trimite_f.php" ce realizează trimiterea efectivă a formularului către o adresă de e-mail, în cazul nostru către: diaconu_lvm@yahoo.com. Din PHP am utilizat funcția *mail*, cu patru parametrii: adresa de e-mail a destinatarului ($catre), subiectul mesajului ce va fi trimis ($subiect), conținutul mesajului ($email) şi detalii ce vor apare în zona de header a mail-ului ($de_la).

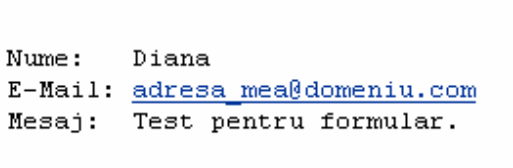
```
<?php
$email = "Nume:\t$_POST[nume]\n";
$email                .=              "E-
    Mail:\t$_POST[email]\n";
$email                                .=
    "Mesaj:\t$_POST[mesaj]\n\n";
$catre = "diaconu_lvm@yahoo.com";
```

Mesajul ce apare în browser, după ce utilizatorul a dat click pe butonul "Trimiteti !"

```
$subiect = "Mesajul trimis";
$de_la = "From: Formular\n"."Reply-
    To: $_POST[email]\n\n";
mail($catre,    $subiect,    $email,
    $de_la);
echo    "<html><head><title>Formular
    trimis!</title></head><body>";
echo "<center>";
echo "Mesajul a fost trimis!";
echo "</center>";
echo "</body></html>";
?>
```



Mesajul ce apare în e-mail-ul primit la adresa de e-mail setată, în cazul nostru: "diaconu_lvm@yahoo.com"



Observații Exemplul de mai sus funcționează şi a fost testat pe un server Apache 1.3.34, ce suportă PHP 4.3.11. În plus, pentru a trimite un e-mail, trebuie să avem instalat pe computer un server de e-mail. Un exemplu de server de e-mail este QK SMTP Server, pe care îl putem găsi la adresa: http://www.qksoft.com/download.html.

Conectarea la MySQL

Pentru a lucra cu o bază de date precum MySQL, trebuie să stabilim o conexiune la serverul bazei de date. Această conexiune este utilizată ca punct de acces pentru orice comenzi ulterioare. Sintaxa este:

$NumeConexiune=mysql_connect('NumeServer', 'NumeUtilizator', 'parola')

NumeServer – este opțional şi specifică serverul la care suntem conectați. Valoarea inițială este "localhost"

Exemplu În următorul exemplu se testează dacă ne putem conecta la baza de date

```
<?php
$x = mysql_connect("localhost","Diana","parola");
if (!$x)
  {
  die('Nu va puteti conecta: ' . mysql_error());
  }
?>
```

După ce am încheiat lucrul cu baza de date, ar trebuie să închidem sesiunea cu mysql_close()