Limbajul PHP (Hypertext Preprocessor, 1994)

Introducere

Ce ne trebuie pentru a lucra in PHP?

Pentru a putea incepe programarea in limbajul PHP, aveti nevoie de urmatoarele:

- un server web ce dispune de interpretorul PHP
  (cel mai simplu este sa va instalati singuri un web-server, pe calculatorul personal)
- de un editor text.
- sa salvati **toate** scripturile pe care le creati in locatia speciala numita **Document Root**

  Este important ca fisierele sa fie salvate intr-un anumit folder de unde sa poata fi preluate de serverul web

- de un browser web (Firefox, Internet Explorer, Chrome, Opera, Safari, etc)

Ce reprezinta un server?

Internet-ul poate fi descries ca acea colectie uriasa de echipamente legate intre ele in scopul schimbului de informatii sau al furnizarii de servicii. Daca ar fi sa simplificam mult ideea de Internet, am putea considera ca acesta este o retea de calculatoare in care fiecare nod (calculator) gazduieste informatii sau servicii ce pot fi accesate de publicul larg. Aceste calculatoare din Internet poarta numele de **servere**.

In sens larg, un **server** este un dispozitiv (combinatie de hardware si software) care ofera servicii si/sau informatii utilizatorilor (clientilor).

Odata cu notiunea de server apare si cea de **arhitectura client-server**, care se refera la un ansamblu format dintr-un dispozitiv server (furnizor de informatii) si un dispozitiv (calculator) client, conectate prin intermediul unei retele si care fac schimb de informatii.

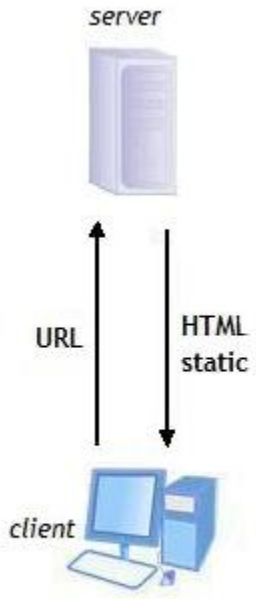
Un tip particular de server, este **server-ul web**. Un server web este un sistem care gazduieste si ofera pagini web prin intermediul unei retele. De cele mai multe ori, termenul de *server web* desemneaza a aplicatie, un program care ruleaza pe calculatorul-server si care este responsabil cu primirea cerintelor de la utilizatori si transferarea paginilor web catre ei.

Paginile stocate pe un calculator-server si oferite publicului larg sunt grupate sub denumirea generica de **site**. Un site (denumit si **website**) este, asadar, o colectie de pagini web, interconectate, stocate pe un server web.

Deservirea paginilor web

In momentul in care un server web primeste o cerinta de la un utilizator pentru o pagina, se verifica mai intai existenta acesteia. Daca pagina exista fizic pe server atunci este transmisa catre utilizator. Paginile returnate de un server web sunt de obicei in format HTML. Navigatoarele web (browserele) sunt capabile sa interpreteze codul HTML si sa afiseze informatiile intr-un mod usor de citit.

In imaginea de mai jos este reprezentata o comunicare intre un server web si un client.



Deservirea unei pagini statice, fara interventia interpretorului PHP

In lumea reala, paginile deservite de un server web sunt de cele mai multe ori **modificate** inainte de a fi transmise catre clienti. Exista situatii cand paginile cerute nici nu exista fizic pe calculatorul server si cu toate astea ele sunt construite si deservite la cerere.

Acest lucru este posibil gratie unor module sau aplicatii aditionale care functioneaza impreuna cu aplicatia de server web. Unul din aceste module este **PHP**.

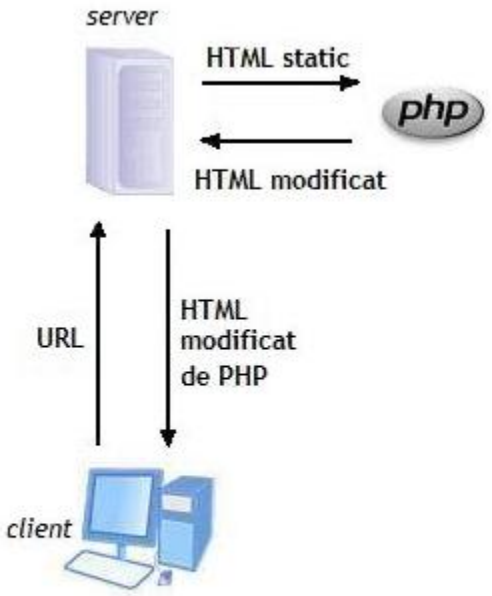
Programarea web

PHP este un limbaj de programare de tip interpretat. Acest lucru inseamna ca fisierele ce contin cod-sursa PHP sunt interpretate ca atare in momentul executiei, de catre PHP. Asadar, pentru executia unei portiuni de cod PHP este folosit codul-sursa asa cum a fost scris el, si nu este transformat intr-o forma intermediara (binara sau cod-masina) cum se intampla la Java sau C/C++. Acest lucru ofera flexibilitate, intrucat orice modificare a fisierelor sursa va fi aplicata imediat la urmatoarea executie, fara alti pasi intermediari. Exista si dezavantaje la acest mod de lucru, cum ar fi timp mai mare de executie a codului, dar in anumite situatii avantajele pot cantari mai mult decat dezavantajele. Datorita faptului ca limbajul este unul interpretat, PHP mai este numit si **limbaj de scripting**.

In sens mai larg, PHP este un limbaj de programare universal, oferind toate facilitatile oricarui limbaj avansat. Codul scris in PHP poate face aproape aceleasi lucruri ca un cod de C/C++ sau Java. Cu toate astea, PHP s-a impus in zona web, ca limbaj server-side, ce extinde functionalitatea serverelor web. Din acest motiv programarea in PHP mai este denumita si **programare web** sau **programare web server-side**.

PHP si paginile web dinamice

Interpretorul PHP actioneaza ca o componenta aditionala, o extensie a serverului web care este invocata de ori de cate ori o pagina PHP este accesata. Aceasta componenta proceseaza codul-sursa din pagina si apoi transmite rezultatul inapoi la web-server, ajungand in final in browserele utilizatorilor.



Deservirea unei pagini dinamice, modificata de PHP in momentul request-ului

Static si dinamic

Din imaginile de mai sus, se observa ca atunci cand nu exista un interpretor PHP, paginile sunt transmise direct catre utilizatori asa cum sunt salvate pe disc, fara modificari. Pentru a actualiza continutul acestora, este nevoie de interventie directa asupra lor si salvarea modificarilor pe server. Aceste pagini sunt denumite**"pagini statice"**.

Spre exemplu, presupunand ca avem o pagina statica ce afiseaza membrii unei comunitati, la fiecare inscriere a unei noi persoane, pagina ar trebui modificata manual de catre cineva cu acces la serverul web. Lucrurile se complica daca acea lista este personalizata, cu trimiteri catre alte informatii (cum ar fi detalii de contact pentru fiecare, etc) sau cu un design intortocheat. Toate aceste probleme pot fi rezolvate cu ajutorul PHP.

Folosind o secventa de cod PHP am putea prelua lista de membri dintr-o baza de date, eliminand problema actualizarii - nu va mai fi nevoie sa se modifice pagina odata cu fiecare membru nou, scriptul PHP va afisa in mod automat noile persoane adaugate in baza de date. Este rezolvata si problema linkurilor personalizate, sau a designului - toate elementele specifice unei persoane pot fi generate in mod automat.

Aceste pagini sunt, asadar, modificate de catre PHP la momentul accesarii lor de catre utilizatori. In functie de parametrii primiti si de secventa de cod definita de programator, aceasi pagina poate avea continut diferit. Aceasta proprietate este denumita dinamism, iar o astfel de pagina este considerata **pagina dinamica**.

EasyPHP

Toate scripturile (programe PHP) vor fi memorate in subfolder-ul *www* al folder-ului *EasyPHP* din *Program Files.*

Scripturi-le vor fi apelate dintr-un browser folosind adresa: "https://localhost:8888/numescript.php".

Orice script va avea extensia *.php*

Exemplu de script php (*primul.php*):

```
<?php
echo "Eu sunt <b> primul</b> <br>script <b>php</b>";
?>
```

Scriptul va fi apelat dintr-un browser folosind adresa: http://localhost/primul.php sau http://localhost:8888/primul.php

Rezulatul va fi o pagina web cu continutul
Eu sunt **primul**
script **php**

C:\Program Files (x86)\EasyPHP-5.3.9\home\en\index.html

Formulare

Formularele sunt elemente ale limbajului HTML, care permit transmiterea datelor si a comenzilor catre server. Pe server, comenzile sau datele sunt receptionate de un script PHP care raspunde comenzilor si prelucreaza datele.
Un formular se descrie intre tag-urile <FORM action *script*> si </FORM>
Intre cele doua tag-uri se insereaza diferite componente, care permit introducerea datelor.

INPUT – element HTML de mare complexitate. El permite utilizatorului sa introduca date si comenzi in pagina furnizata de browser.

Forma generala:

<INPUT type="tip" value="valoare" align="tip_aliniere">

Tipurile elementului INPUT le vom prezenta in sectiunile urmatoare.

Tipul "submit"

Acest tip are aspectul unui buton. La apasarea butonului submit datele introduce sunt trimise catre server, de unde, raspunde un script care are rolul de a le prelucra.

Exemplu:

Fisier HTML (**pag2.html**):

```
<html>
<head></head>
<body>
<form action="http://localhost:8888/apl2.php">
<input type="submit" value="Primul exemplu de apel PHP">
</form>
</body>
</html>
```

Fisier PHP (**apl2.php**):

```
<?php
echo ("Ai apasat, am raspuns");
?>
```

Tipul "text"

Este de forma unui edit si permite utilizatorului sa introduca date. Acestea se trimit catre server dupa apasarea unui buton de tip submit.

Exemplu:

pag3.html

```
<html>
<head></head>
<body>
<form action="http://localhost:8888/apl3.php" method="post">
a= <input type="text" name="a"><br>
b= <input type="text" name="b"><br>
<br>
<input type="submit" value="Suma?">
</form>
</body>
</html>
```

apl3.php

```
<?php
//recuperarea datelor
$a = $_POST['a'];
$b = $_POST['b'];
echo "suma= ", $a+$b;
?>
```

Operatori

Operatorii PHP in ordinea descrescatoare a prioritatii:

1. !, ++, --, (int), (double), (string);
2. *, %, / ;
3. <, <=, >, >=;
4. == ,!= ,===, !==;
5. &;
6. ^;
7. ^^;
8. ?:;
9. =, +=, -=, /=, *-, %=, &=, |=, ^=;
10. and;
11. xor;
12. or;
13. ,.

Exemple:

```
<?php

$x=1;

$y=7.5;

$x=(int)$y; //partea intreaga din$y

echo $x;

?>
```

```
<?php

$x=”-2.76”;

$y=”1.76”;

echo  ($x+$y);

?>
```

Instructiunile limbajului PHP

Exemplu de fisier HTML pentru citirea unui numar “x”:

```
<html>
<head>
</head>
<body>
    <form action=”http://localhost:8888/script.php” method=”post”>
        Formular de introdus date, pentru a fi prelucrate cu un script <br><br><br>
        x = <input type=”text” name=”x”> <br><br>
        <input type=”submit” value=”Trimite”>
    </form>
</body>
</html>
```

Instructiunea expresie

Se foloseste pentru atibuiri.

Exemplu:

```
$x = $y*$y+1;
```

Instructiunea if

Forma 1:

```
if (expresie_logica)
```

instructiune1;

Forma 2:

if (expresie_logica)

instructiune1;

else

instructiune2;

Daca expresia este adevarata atunci se executa instructie1 si nu se executa intructiune2, iar daca expresia este falsa atunci se executa intructiune2, daca exista si nu executa intructiune1.

Exemplu:

script.php

<?php

$x=$_POST['x'];

if($x>=0)

echo "x este pozitiv";

else

echo "x este negativ";

?>

Instructiunea compusa si instructiunea alternative switch

Atunci cand dorim ca mai multe instructiuni sa fie interpretate ca una singura, instructiunile se trec intre "{" si "}".

Instructiunea **switch** are forma generala:

switch(*expresie*){

case **exp1**: *secventa instructiuni 1*; break;

case **exp2**: *secventa instructiuni 2*; break;

…

case **expn**: *secventa instructiuni n*; break;

[default: *secventa instructiuni n+1*];

}

Exemplu: Pentru x avand valoarea 1,2,3,4. Valoarea respectiva este retransmisa catre utilizator alfabetic, altfel se va transmite un mesaj oarecare.

```
<?php
$x=$_POST['x'];
switch($x){
    case(1): echo("unu"); break;
    case(2): echo("doi"); break;
    case(3): echo("trei"); break;
    case(4): echo("patru"); break;
    default: echo("Nu este 1,2,3 sau 4");
}
?>
```

Instructiunea for

Instructiunea **for** are forma generala:

for(*expresie initiala*; *expresie test*; *expresie incrementare*)

Instructiune

Exemplul 1: Suma patratelor primelor x numere naturale (x numar natural).

```
<?php
    $x=$_POST['x'];
    $s=0;
    for($i=1;$i<=$x;$i++)          //i++ inseamna i=i+1
        $s += $i*$i;
```

```
        echo “suma patratelor numerelor <= “,x,” este “,$s;
?>
```

Exemplul 2: Afisati toate numerele prime <=x. Pentru x=10 se va afisa 2 3 5 7.

```
<?php
        $x=$_POST[‘x’];
        for($k=2; $k<=$x; $k++){
                //verif daca $k este prim
                $sw=1;
                for($i=2; $i<=$k/2;$i++)
                        if($k % $i ==0)
                                $sw=0;
                if($sw==1)
                        echo($k.” ”); //concatenare de siruri de caractere
        }
?>
```

Instructiunea while

Instructiunea **while** are forma generala:

while(*expresie*)

Instructiune

Exemplu: Suma de la exemplul anterior.

```
<?php
        $x=$_POST[‘x’];
        $s=0; $i=1;
        while($i<=$x){
```

```
        $s += $i*$i;
        $i++;
    }
    echo($s);
?>
```

Instructiunea do while

Aceasta instructiune are forma generala:

```
do
    Instructiune
while (expresie);
```

Exemplu: Suma de la exemplul anterior.

```
<?php
    $x=$_POST['x'];
    $s=0; $i=1;
    do{
        $s += $i*$i;
        $i++;
    }while($i<=$x);
    echo($s);
?>
```

1. Calculati suma cifrelor lui $x. Pentru $x=2014 se va afisa 7.
2. Afisati cifrele lui x una sub alta in ordine inversa pe randuri diferite.

Tipul “radio”

Tipul **radio** (butoane radio) se utilizeaza in cazul in care utilizatorul trebuie sa aleaga o singura valoare, din mai multe variante posibile, dorindu-se afisarea tuturor posibilitatilor, separate. Pentru aceasta trebuie sa avem un element input cu un acelasi nume (**name**), valori diferite (**value**), de tip radio, putand avea, optional, o valoare preselectata (**checked**) . In exemplul de mai jos, a doua valoare este preselectata. Odata ce o valoare a fost selectata, in cazul in care nici o valoare nu era selectata prestabilit (adica input-ul nu continea comanda *checked*), este obligatoriu sa avem selectata o valoare. Se recomanda folosirea comenzii *checked*, pentru a preveni posibilele erori cauzate de lipsa unei valori selectate.

```
<FORM action=”http://localhost/exemplu.php” method=”post”>
<p>      Cumpar ziarul … </p>
<br><br>
<INPUT name=”Buton” TYPE=”radio” value=1> zilnic <BR>
<INPUT name=”Buton” TYPE=”radio” value=2 checked> saptamanal<BR>
<INPUT name=”Buton” TYPE=”radio” value=”3”> lunar<BR>
<INPUT name=”Buton” TYPE=”radio” value=”4”> anual<BR>
<INPUT TYPE=”submit” value=”Trimite!”>
</form>
```

Tipul “checkbox”

Tipul **checkbox** se utilizeaza in cazul in care utilizatorul trebuie sa aleaga din mai multe variante posibile, valorile pe care le doreste. Deosebirea dintre tipul radio si tipul checkbox consta in faptul ca, in cazul tipului checkbox, putem avea selectate mai multe valori, in acelasi timp, ori sa avem toate valorile deselectate. Dar, in cazul acestui tip, se transmit catre server **doar valorile butoanelor marcate**, nu si numele acesora. Deasemenea, si in acest tip putem folosi comanda **checked**, in acelasi mod, doar ca putem folosi aceasta variabila pentru mai mult de o valoare. In exemplul de mai jos, valorile preselectate sunt prima si ultima.

```
<form action=http://localhost/exemplu.php” method=”post”>
<P>      Urmaresc postul tv </P>
<INPUT name=”buton1” TYPE=”checkbox” value=1 checked> Realitatea  <BR>
<INPUT name=”buton2” TYPE=”checkbox” value=2 > Antena 1 <BR>
<INPUT name=”buton3” TYPE=”checkbox” value=3 > Antena 2  <BR>
<INPUT name=”buton4” TYPE=”checkbox” value=4 > Antena 3  <BR>
<INPUT name=”buton5” TYPE=”checkbox” value=8 checked> Protv  <BR>
</form>
```

Tipul “password”

Componentele vizuale de acest tip se utilizeaza pentru introducerea parolelor, atat de necesare, de exemplu, pentru a avea acces la un site. Sunt asemanatoare cu cele de tip text, numai ca , pentru fiecare character al parolei se afiseaza cate un “*”. Desigur, catre server, datele sun transmise correct.

Exemplu:

```
<FORM Action=”http://localhost/exemplu.php” Method=”post”>
<P> Introduceti parola </P>
<INPUT NAME=”parola” TYPE=”password” <BR>
<INPUT type=”submit” value=”GO”>
</FORM>


<?php
echo $_POST[‘parola’];
?>
```

Elementul TEXTAREA

Elementul **TEXTAREA** se utilizeaza pentru a introduce un text mai lung. El poate fi folosit in interiorul unui formular si , la apasarea butonului **submit**, **TEXTAREA** se pot preciza numarul de linii (**rows**) si de coloane(**cols**) pe care acesta sa le afiseze.

Exemplu

```
<FORM action=http://localhost/exemplu.php method=”post”>
<P>Adaugati comentariul dv.!</P>
<TEXTAREA name=”textul” rows=”10” cols=”20”></TEXTAREA><BR><BR>
<INPUT type=”submit” value=”Scrie”>
</FORM>

<?php
echo $_POST[‘textul’];
?>
```

Elementul SELECT

Elementul **SELECT** se utilizeaza pentru ca browser-ul sa afiseze o lista. FIecare optiune din lista este marcata de un element **OPTION.** Daca este introdusa intr-un formular, catre server se va trimite optiunea selectata.

Exemplu

<FORM action=http://localhost/testare.php " method="post">
<SELECT name="lista">
<OPTION value="optiunea1">Optiunea 1 </OPTION>
<OPTION value="optiunea 2">Optiunea 2 </OPTION>
<OPTION value="optiunea 3">Optiunea 3</OPTION>
</SELECT>
<INPUT type="submit" value="GO">
</FORM>

<?php
$v=$_POST["lista"];
Echo $v;
?>