1.Variabile predefinite in PHP

$GLOBALS
= pot fi accesate toate variabilele globale care sunt accesibile script-ului PHP curent
$_SERVER = contine o serie de variabile ale caror valori sunt setate de server-ul web; majoritatea valorilor variabilelor din acest vector depind de mediul de executie al script-ului curent.
$_GET si $_POST contin variabile primite de script prin intermediul unor transferuri care folosesc metodele HTTP get, respectiv post. De exemplu, prin intermediul acestor vectori, pot fi accesate valorile campurilor dintr-un formular care a fost completat si transmis folosind una dintre cele doua metode.
$_COOKIE contine valorile variabilelor care cuprind informatii referitoare la cookie-urile pastrate pe calculatorul utilizatorului ce acceseaza pagina web.
$_FILES contine variabile primite de script prin intermediul incarcarilor de fisiere prin metoda post.
$_ENV contine variabile disponibile prin intermediul mediului in care este executat.
$_REQUEST contine variabile disponibile prin intermediul oricarui tip de mecanism cu ajutorul caruia utilizatorul poate introduce date.
$_SESSION contine variabile care corespund sesiunii curente a script-ului.

2.Utilizare formulare HTML cu PHP, $_GET si $_POST

Invatam să preluam date trimise prin GET si POST

1. Receptionarea datelor de la un formular HTML

In general datele din formular sunt preluate de scriptul PHP prin urmatoarea formula:

$_POST['nume'] - daca este folosit method="post"
$_GET['nume'] - daca este folosit method="get"

- unde "nume
este valoarea atributului name al elementului din formularul HTML.

Sa luam un exemplu practic de formular HTML care trimite date (prin method="post") la un script PHP unde acestea vor putea fi vizualizate.
Salvam scriptul de mai jos intr-un fisier pe care-l numim "test-form.php"

```
<?php
$nume = $_POST['nume'];
$email = $_POST['email'];
$parola = $_POST['parola'];
echo "Nume = $nume";
echo "<br />E-mail = $email";
echo "<br />Parola = $parola";
?>
```

Scriem urmatorul cod HTML intr-un alt fisier "form.html", pe care-l salvam in acelasi director cu scriptul PHP de mai sus.

```
<html >
<head>
<title> Test-Form </title>
</head>
<body>
<form action="test-form.php" method="POST">
Nume:<input type="text" name="nume" />
<br />Email:<input type="text" name="email" />
<br />Parola:<input type="password" name="parola" />
<br /><input type="submit" name="submit" value="Trimite datele" />
</form>
</body>
</html>
```

In browser va aparea:

Nume:
Email:
Parola:

Dupa ce am competat datele, apasam clic pe butonul "Trimite datele", acestea vor fi trimise la scriptul PHP "test-form.php", care le va prelucra si va afisa REZULTATELE:

Sa intelegem exemplu de mai sus.

Folosind formularul de mai sus, atributul NAME din fiecare eticheta INPUT atribuie fiecarei casete cu text un nume, astfel scriptul PHP va putea recunoaste datele scrise in casete. In scriptul "test-form".php" vom accesa variabilele:

$nume – va primi informatia introdusa in campul Nume
$email – va primi informatia introdusa in campul Email
$parola – va primi informatia introdusa in campul Parola

Deoarece cunoastem metoda prin care trimitem datele catre scriptul PHP, "POST", am folosit variabila PHP globala _POST pentru a prelua datele din formular:

$_POST['nume']
$_POST['email']
$_POST['parola']

Constructia echo trimite datele de iesire care vor fi afisate de browser

Variabile PHP globale: _GET si _POST reprezinta de fapt variabile de tip array, fiecare element se poate accesa prin cheia sa; in cazul nostru cheia fiecarui element este data de atributul : NAME al casetelor din formular

3. Trimiterea de date unui script prin adresa URL

In afara de a expedia unui script datele printr-un formular, puteti expedia date cu ajutorul adresei URL a paginii. Pentru aceasta, atasati la sfarsitul adresei URL un semn al întrebarii (?) si apoi includeti o serie de perechi "nume-valoare" (separate prin &), ca in exemplu urmator:

- https://adresa paginii/fisier.php?nume1=valoare1&nume2=valoare2

Exemplul include numai doua perechi "nume-valoare"; cu toate acestea, puteti include oricâte asemenea perechi doriti (separate prin caracterul &), in functie de limita impusa de browser.

Pentru a prelua si folosi datele dintr-o astfel de adresa URL, folositi in interiorul scriptului PHP expresia "$_GET['nume'], ca in exemplu urmator"

- $var1 = $_GET['nume1']
- $var2 = $_GET['nume2']

Unde "nume1" si "nume2" sunt numele variabilelor din adresa URL, iar "$var1" si "$var2" sunt variabilele care vor fi folosite in scriptul PHP (din "fisier.php") si a caror valori vor fi "valoare1" respectiv "valoare2" continute in adresa URL.

Daca doriti sa trimiteti unui script, prin intermediul adresei sale URL, caractere speciale precum un semn al intrebarii, un semn egal sau un ampersand, se poate crea confuzie.

Pentru a functiona corect, un sir trebuie sa fie codificat URL. Pentru a codifica URL un sir, caracterele speciale se înlocuiesc cu echivalentele lor hexazecimale, precedate de un simbol procent (%).
De exemplu, forma codificata URL a sirului "la multi ani!" este %22la multi ani%21%22.

Adresa URL rezultanta se numeste "sir de interogare" si nu poate contine spatii. Daca doriti sa trimiteti un spatiu ca parte a unui sir de interogare, trimiteti în locul spatiului un semn plus (+).

Unele dintre cele mai comune caractere speciale si echivalentele lor codificate URL sunt prezentate în tabelul de mai jos:

Caracter special	Echivalentul codificat URL
.	*%2e
>	%3e
^	%5e
~	%7e
+	%2b
,	%2c
/	%2f
:	%3a
;	%3b
	%3c

=	%3d
>	%3e
[	%5b
\	%5c
]	%5d
_	%5f
{	%7b
|	%7c
}	%7dc
tab	%09
spatiu	%20
!	%21
“	%22
#	%23
$	%24
%	%25
&	%26
`	%27
(	%28
)	%29
@	%40
`	%60

)

4. Scrierea instructiunilor "if"

Expresiile conditionale sunt esențiale pentru scrierea instrucțiunilor condiționale, prin care se iau decizii.

Una dintre cele mai simple și folosite instructiuni condiționale este instrucțiunea : if.

```
If(conditie) {
  Instructiune(1);
  Instructiune(2);
  ..................
}
```

- După cum se observă dupa if urmează o pereche de paranteze rotunde în interiorul cărora se plasează condiția, adică o expresie logica a carei rezultat poate fi TRUE sau FALSE. Dacă și numai dacă rezultatul expresiei logice (conditia) este TRUE se vor executa instuctiunile : Instructiune(1), apoi Instructiune(2), etc...

Se observă ca acest set de instructiune se plasează intre acolade. Doar într-un singur caz aceste acolade nu sunt necesare și anume: în cazul în care este scrisă doar o singură instructiune.

Dacă rezultatul expresiei logice dintre paranteze rotunde este FALSE , setul de instructiuni dintre acolade nu se va executa, controlul programului va trece mai departe.
Iata inca un exemplu, practic:

```
<?php
$numar = 12;
if ($numar > 10) {
  echo "Acesta este un numar mai mare decat 10";
}
?>
```

- Atunci când este executat scriptul, instructiunea if evaluează expresia condiționala $numar > 10, care este adevarată (TRUE) numai dacă valoarea variabilei $numar este mai mare decât 10. Dacă valoarea variabilei $numar este mai mare decat 10, se va executa instrucțiunea echo, în caz contrar programul va trece mai departe.

In general, limbajul PHP ignoră spațiile albe. In mod conventional, o instrucțiune asociată unei instructiuni "if" este scrisa decalat în raport cu aceasta. Acest procedeu este recomandat deoarece prin utilizarea sa este facilitată citirea programului.

Utilizarea instructiunii "else"

Să presupunem ca doriți să executati o instrucțiune atunci când o conditie este TRUE și o altă instrucțiune când conditia este FALSE. Instrucțiunea else vă permite să procedați astfel, după cum urmează:

```
If(conditie)
{
  Instructiune(1);
  Instructiune(2);
  .....................
}
else
{
  Instructiune(3);
  Instructiune(4);
  .....................
}
```

In acest din urma caz daca expresia logica din paranteze: conditie este adevarata se vor executa instructiunile: Instructiune(1); Instructiune(2); , iar daca valoarea expresiei logice : conditie are valoarea FALSE se vor executa instructiunile: Instructiune(3); Instructiune(4);

Instructiunea asociata unei instructiuni if sau else poate fi ea însasi o instructiune if. O asemenea instructiune if se numeste "instructiune if imbricata".
Iata un exemplu de instructiune if imbricata:

```
<?php
$numar = 88;
if ($numar > 10) {
  if ($numar > 100) {
    echo "Acesta este un numar mai mare decat 100";
  }
  else {
    echo "Acesta este un numar mai mic decat 100, dar mai mare decat 10";
  }
}
else {
  echo "Acesta este un numar mic";
}
?>
```

Instructiunile if imbricate pot deveni extrem de dificil de înteles daca numarul de instructiuni si nivelul de imbricare nu sunt relativ reduse. Deci trebuie sa le folositi cu economie.

O instructiune corelata atât cu instructiunea if, cât si cu instructiunea else, este instructiunea elseif. Când este folosita corect, poate fi mai simplu de înteles decât o instructiune if imbricata, logic echivalenta cu aceasta.
Iata un exemplu de instrutiune else if():

```
<?php
$numar = 88;
if ($numar > 100 ) {
  echo "Acesta este un numar mai mare decat 100";
}
elseif ($numar > 10) {
  echo "Acesta este un numar mai mic decat 100, dar mai mare decat 10";
}
elseif ($numar > 1) {
  echo "Acesta este un numar mic";
}
else {
  echo "Acesta este un numar foarte mic";
}
?>
```

Exemplul extinde functionalitatea exemplului anterior, afisând mesajul "Acesta este un numar foarte mic" pentru valori ale variabilei $numar mai mici sau egale cu 1.

Într-un caz general, cu o instructiune if si cu o instructiune else poate fi asociat un numar mult mai mare de instructiuni elseif. PHP evalueaza expresiile conditionale în mod succesiv, pornind de la expresia conditionala asociata instructiunii "if". PHP executa instructiunea asociata primei expresii conditionale care are valoarea TRUE; daca nici o expresie conditionala nu are valoarea TRUE, PHP executa instructiunea asociata cu instructiunea else. Este permisa omiterea instructiunii else, caz în care nu

este executata nici o instructiune daca nici una din expresiile conditionale nu are valoarea TRUE.

Scrierea instructiunilor switch, break si default

In cazul in care vrem sa comparam valoarea unei singure variabile cu o succesiune de valori, in locul instructiunii "if" putem folosi instructiunea switch:

De exemplu, sa presupunem ca valoarea variabilei $numar este l, 2 sau 3, reprezentând respectiv dimensiunile "mica", "medie" si "mare". Iata un mic script care afiseaza dimensiunile asociate valorilor variabilei $numar, folosind functia switch alaturi de break si default care vor fi explicate mai jos:

```
<?php
switch($numar)
{
    case(1):
      echo "mic";
      break;
    case(2):
      echo "mediu";
      break;
    case(3):
      echo "mare";
      break;
    default:
      echo "Acesta nu este un cod valabil";
}
?>
```

- Actiunea unei instructiuni switch este determinata de valoarea unei expresii întregi, nu de valoarea unei expresii conditionale. Numele variabilei este dat între parantezele care urmeaza dupa cuvântul cheie switch. Acoladele delimiteaza o serie de instructiuni case si o instructiune default optionala, fiecare dintre instructiunile cuprinse între acolade putând avea instructiuni asociate.

Când este executata, instructiunea switch încearca sa stabileasca o identitate între valoarea variabilei sale si valoarea asociata unei instructiuni case. Se vor executa instructiunile asociate primei instructiuni "case" pentru care identitatea respectiva este valabila.

Daca valoarea variabilei din instructiunea switch nu corespunde nici uneia din valorile asociate instructiunilor case, se vor executa instructiunile asociate instructiunii default, daca exista o asemenea instructiune (ne este obligatorie).

Un procedeu de programare indicat consta în aceea ca fiecare instructiune case din cadrul unei instructiuni switch sa se încheie cu o instructiune break.

; Instructiunea break determina încheierea executiei instructiunii switch, sare peste "case-urile" ramase si se executa codurile care mai sunt (daca exista) dupa "switch". În absenta instructiunii "break", executia trece la urmatoarea instructiune "case" sau "default", fapt nedorit în majoritatea cazurilor.

• Nu este necesar sa folositi numere întregi consecutive în instructiunile case ale unei instructiuni switch. Daca preferati, puteti folosi numere întregi non-consecutive,

numere cu virgula sau siruri. Valoarea de la "case" se poate adauga si fara paranteze, separata prin spatiu.

- Exemplu:

```
<?php
$site = 'UPG';
switch($site) {
  case 1:
    echo 'php.net';
    break;
  case 'coursesweb':
    echo 'http://coursesweb.net';
    break;
  case 'UPG':
    echo 'www.upg-ploiesti.ro';
    break;
  default: echo 'google.com';
}
?>
```

4. Operatorul ? :

Operatorul conditional ?:, denumit uneori "operator ternar" sau "operator întrebare-doua puncte", constituie o alta modalitate de a scrie decizii în PHP.

Acest operator formeaza o expresie care se poate folosi în multe contexte PHP. Iata sintaxa de utilizare a acestuia:

expresie-conditionala ? valoare-TRUE : valoare-FALSE

Observati cum semnul întrebarii este separat de caracterul doua puncte prin valoarea valoare-TRUE.

Operatorul conditional îsi evalueaza expresia conditionala. Daca expresia este evaluata la valoarea TRUE, operatorul conditional returneaza valoarea valoare-TRUE; în caz contrar, returneaza valoarea valoare-FALSE.

De exemplu, sa luam în considerare urmatoarea instructiune de atribuire, care foloseste un operator conditional:

```
$a = ($b > $c) ? 10  : 20;
```

Aceasta instructiune de atribuire compara valorile variabilelor $b si $c. Daca valoarea variabilei $b este mai mare decât aceea a variabilei $c (adica TRUE), atunci variabilei $a îi este atribuita valoarea 10; în caz contrar, variabilei respective îi este atribuita valoarea 20.