1.SESIUNI IN PHP

Când un utilizator răsfoiește paginile unui site, de cele mai multe ori are nevoie de un mecanism care să-i permite accesul în anumite informații, indiferent de pagina pe care o traversează.

Mecanismul este următorul:în momemtul când utilizatorul accesează pagina unui site care beneficiază de o astfel de facilitate, se inițiază o sesiune care are ca efect generarea aleatoare a unui identificator unic de sesiune(ID).

In PHP o sesiune reprezinta perioada de timp in care mai multe scripturi PHP, accesate la momente diferite de timp, pot stoca si folosi informatii comune. O sesiune incepe atunci cand un script apeleaza functia **session_start** si se termina atunci cand utilizatorul inchide browserul (exista si alte modalitati de a porni o sesiune, dar nu sunt prea uzuale - folosirea comenzii session_start este metoda recomandata).

O sesiune se intinde pe mai multe requesturi (pe parcursul a mai multor accesari ale diferitelor pagini), iar pentru a identifica existenta unei sesiuni PHP poate folosi cookie-uri sau parametrii GET un URL-ul paginii.

Cookies reprezinta portiuni de informatii (stocate sub forma de fisiere de mici dimensiuni) ce se afla pe calculatorul utilizatorului si care sunt create si folosite de catre browser in comunicarea cu serverul web. De obicei cookie-urile sunt folosite pentru a identifica utilizatorii sau a pastra urma vizitelor pe un site.

Cookie-urile pot fi sterse cu usurinta de catre utilizator, sau pot fi blocate de catre browser, deci folosirea lor trebuie facuta cu grija si doar in cazuri de necesitate. Un cookie poate contine o cantitate limitata de informatie iar durata de viata poate fi limitata (la un anumit numar de zile, la inchiderea sesiunii de lucru, etc) sau nelimitata (pana la stergerea lor).

PHP dispune de 2 functii prin care se pot crea cookie-uri: setcookie si setrawcookie. Cookie-urile create pe calculatorul utilizatorului pentru un site sunt transmise de catre browser inapoi la server si sunt disponibile in variabila globala **$_COOKIE.**

In momentul in care un script apeleaza functia **session_start** pentru prima data intr-o sesiune de lucru, se transmite un cookie catre browserul clientului (un header de tipul 'Set-Cookie'). Fiind vorba de un cookie, este necesar ca functia **session_start** sa fie apelata inaintea oricarei instructiuni ce afiseaza ceva (**print, echo, etc**) si inaintea oricarui cod HTML. Cookie-ul transmis contine un identificator ce poarta numele de **Session ID**, pe baza caruia se poate face distinctie intre sesiunea curenta si alte sesiuni ale altor utilizatori ce acceseaza site-ul in acel moment.

In cazul in care browserul utilizatorului nu accepta cookie-uri, identificatorul de sesiune va fi transmis printr-un parametru GET, in forma script.php?PHPSESSID=[session id] (se va face practic un redirect automat la aceeasi pagina avand specificat parametrul in URL). Este apoi responsabilitatea programatorului sa includa manual acest identificator in toate celelalte link-uri de pe pagina, asigurandu-se ca toate paginile vor fi accesate cu acest parametru. Aceste situatii sunt insa rare, iar in exemplele ce urmeaza vom considera ca browserele au mereu cookie-urile activate, astfel ca nu va trebui sa avem grija sa transmitem manual Session ID-ul.

In momentul in care se acceseaza din nou aceeasi pagina, sau o alta din cadrul aceluiasi site, identificatorul de sesiune este transmis de catre browser (ca orice cookie existent in browser). Astfel, orice script PHP are acces la Session ID-ul creat initial, fiind capabil sa acceseze sesiunea corecta. Mai este nevoie de ceva insa: pentru a putea avea acces la informatiile

persistate, un script trebuie sa apeleze session_start. De aceasta data, existand deja un Session ID disponibil, PHP va sti ca nu trebuie creata o sesiune noua ci continuata una existenta. Asadar, session_start are doua functionalitati: sa porneasca o sesiune noua (atunci cand nu exista un Session ID) sau sa continue o sesiune existenta, identificata printr-un Session ID.

Functiile principale de manipulare a sesiunii sunt prezentate mai jos. Exista o functie care returneaza Session ID-ul curent: session_id. Aceasta este utila cand este nevoie ca identificatorul sa fie transmis in URL. Alternativ se poate folosi constanta globala SID. De asemenea exista o functie pe permite inchiderea unei sesiuni pe parcursul executiei scriptului curent. Inchiderea sesiunii inseamna oprirea posibilitatii de a scrie/citi date, nu stergerea datelor salvate deja. Datele raman salvate si pot fi accesate din nou dupa apelarea functiei session_start.

```
<?php

session_start();

 echo session_id();

 echo SID; // are acelasi efect ca instructiunea anterioara

 session_write_close(); // permite inchiderea sesiunii in scriptul curent

?>
```

Accesarea datelor

Din momentul in care scriptul PHP apeleaza session_start, acesta poate incepe deja sa stocheze date ce vor fi persistate. In limbajul programatorilor de PHP se foloseste expresia "sa pastreze date in sesiune" sau "pe sesiune". Aceste date sunt gestionate de catre limbajul PHP (salvate, preluate, etc) si nu este esential pentru programator sa cunoasca mecanismul intern de manipulare a acestora.

Salvarea datelor pe sesiune se face prin intermediul vectorului super-global $_SESSION. Exista si o functie ce permite inregistrarea datelor pe sesiune (nu si modificarea lor), dar folosirea acesteia nu este recomandata. Functia se numeste session_register si a fost marcata ca invechita in versiunea 5.3 a limbajului PHP. Citirea datelor persistate se poate realiza tot prin intermediul $_SESSION.

```
<?php

session_start();

 // inregistrarea datelor in sesiune

$_SESSION[ 'text' ] = 'Mesaj persistat';

 // citirea din sesiune

echo $_SESSION[ 'text' ];
```

```
// pe sesiune se pot inregistra aproape orice tipuri de date

$vector = array('a', 'b', 'c');

$_SESSION[ 'litere' ] = $vector;

// accesez o parte din vectorul stocat

echo $_SESSION[ 'litere' ][0];  // afiseaza a

echo $_SESSION[ 'litere' ][2];  // afiseaza c

?>
```

Aplicatie: sa se transmita o informatie de la o pagina la alta folosind sesiunile PHP

Urmand exemplul de la inceputul paginii, sa construim cele doua fisiere a.php si b.php pentru a putea pastra textul de la o accesare la alta.

In primul fisier stochez date pe sesiune, urmand ca acestea sa fie citite de pe sesiune in al doilea fisier.

```
<?php

// a.php

session_start(); // pornesc sesiunea pentru a putea stoca o variabila

$_SESSION[ 'text' ] = 'Mesaj de pe prima pagina';

echo '<a href="b.php">Mergeti la pagina urmatoare</a>';

?>
```

Dupa accesarea paginii a.php ar trebui sa existe deja o sesiune creata ce are stocata variabila text. Scriptul din b.php va accesa aceasta sesiune, tot prin session_start si va citi datele inregistrate anterior.

```
<?php

// b.php

session_start(); // continuam o sesiune existenta

echo $_SESSION[ 'text' ]; // afisam informatia

?>
```

PHP, ca orice alt limbaj modern de programare, ofera facilitati avansate in lucrul cu fisierele. Astfel, folosind cod PHP se pot crea, modifica, manipula si sterge fisiere.

Fisiere text si fisiere binare

Fisierele sunt colectii de informatii stocate pe un dispozitiv de stocare (hard-disk, CD, etc). In functie de formatul datelor continute fisierele se impart in doua categorii: fisiere text si fisiere binare.

Fisierele text sunt cele care contin text simplu ce poate fi citit, reprezentat si modificat de catre oricine si de catre orice secventa de cod. Spre exemplu, un fisier cu extensia .txt, creat cu un utilitar cum este Notepad, este cel mai definitoriu tip de fisier text. Un alt exemplu este un cod-sursa PHP salvat intr-un fisier. Desi codul sursa PHP are o anumita structura si poate fi interpretat intr-un anumit mod, el este la baza un fisier text, ce poate fi citit de oricine (orice om). De asemenea continutul sau poate fi modificat oricand, fie de un om, fie de un program.

Fisierele binare sunt acele fisiere ce contin secvente de text intr-un anumit format si cu o structura specifica inteleasa doar de un calculator (de o anumita secventa de cod). Un exemplu de fisier binar este o imagine (fisier cu extensia .jpg) sau o melodie (fisier cu extensia .mp3). Desi acestea pot fi vizualizate cu un editor de texte, ca Notepad, continutul lor nu poate fi modificat de oricine (cel putin nu fara a corupe fisierul). Fisierele binare sunt utile pentru ca pot contine orice tip de date, cu orice structura se doreste. Avantajul este ca prin intermediul fisierelor binare si pot crea formate proprietare, care permit modificarea continutului fisierelor doar de anumite programe. Spre exemplu, fisiere PDF pot fi create/vizualizate doar cu anumite aplicatii, la fel si documentele Word, s.a.

PHP ofera suport pentru ambele tipuri de fisiere: text si binare. Atat timp cat se cunoaste structura fisierelor binare (ce tipuri de informatii contin) acestea pot fi modificate din PHP. Cu toate acestea, in cele ce urmeaza ne vom concentra asupra fisierelor text intrucat sunt cel mai des intalnite si mai usor de folosit.

Folosirea fisierelor in PHP

In orice limbaj de programare, si deci si in PHP, cand se lucreaza cu fisiere trebuie efectuate urmatoarele operatiuni:

deschiderea fisierului

citirea din fisier/scrierea in fisier

inchiderea fisierului

Necesitatea fiecarei operatiuni este probabil usor de inteles. Un fisier trebuie deschis pentru a putea vedea ce este in el - chiar si pentru o secventa de cod aceasta este o necesitate. Scrierea/citirea propriu-zisa reprezinta operatia pentru care fisierul a fost deschis, prin care sunt preluate sau adaugate informatii in fisier. La finalul folosirii fisierului acesta trebuie sa fie inchis, altfel este posibil ca informatiile continute sa se piarda.

Instructiunile PHP corespunzatoare operatiilor de mai sus sunt prezentate in cele ce urmeaza. De remarcat este faptul ca trebuie specificat inca de la deschiderea fisierului ce fel de operatii vor avea loc: citire, scriere sau ambele.

```
<?php

# deschiderea unui fisier si citirea din el
$id_fisier = fopen("c:\\folder\\fisier.txt", 'r');   // deschidere
fread( $id_fisier, 10 );   // citire 10 octeti din fisier
fclose($id_fisier);   // inchidere
```

```
# deschiderea unui fisier, stergerea continutului si scrierea in el
$id_fisier = fopen("c:\\folder\\fisier.txt", 'w');   // deschidere
fwrite( $id_fisier, 'Text nou in fisier' );
fclose($id_fisier);   // inchidere


# deschiderea unui fisier si scrierea la sfarsitul lui
$id_fisier = fopen("c:\\folder\\fisier.txt", 'a');   // deschidere
fwrite( $id_fisier, 'Text adaugat in fisier' );
fclose($id_fisier);   // inchidere

?>
```

Asa cum s-a observat, un mod de deschidere pentru citire ('r') si 2 moduri de deschidere pentru scriere: cu stergere a continutului vechi (modul 'w') si cu pastrare a continutului (modul 'a'), caz in care scrierea se face la sfarsitul fisierului, imediat dupa continutul vechi. Depinde de specificul fiecarei situatii ce mod de scriere trebuie ales.

Limbajul PHP dispune si de alte moduri de deschidere a fisierului, care sunt mici variatii ale celor prezentate mai sus. Intrucat sunt folosite doar in anumite situatii nu vor fi prezentate aici. Pentru detalii puteti consulta pagina de documentatie a functiei fopen.

Instructiunile de mai sus pot fi folosite cu orice tip de fisier, text sau binar si reprezinta baza in lucrul cu fisierele. Exista o suita de alte functii pentru citirea sau scrierea anumitor tipuri de date specifice, ce pot fi folosite cu ambele tipuri de fisiere. In practica insa, de cele mai multe ori se lucreaza cu instructiuni "utilitare" de manipulare a fisierelor text - instructiuni care simplifica efortul programatorului in special in situatii generale. Aceste functii sunt descrise in cele ce urmeaza.

2. FIȘIERE IN PHP

2.1 Instructiuni pentru fisiere text

Pentru simplificarea codului de citire/scriere a unui fisier in situatii generale, PHP ofera cateva functii foarte convenabile: file_get_contents, file_put_contents si file. Primele doua, dupa cum si numele spune, permit preluarea intregului continut al unui fisier si punerea lui intr-o variabila string, printr-un singur apel, respectiv crearea unui fisier care sa contina valoarea unei variabile (exemple mai jos). Cea de-a treia functie permite crearea unui vector ce are ca elemente liniile fisierului specificat. De exemplu, un fisier cu 3 linii va genera un vector cu 3 elemente, fiecare element reprezentant o linie din fisier.

Este important de mentinut ca odata citit continutul fisierului (cu file_get_contents si file) variabilele se "deconecteaza" de fisier, adica orice modificare facuta asupra variabilei nu se reflecta si asupra fisierului, cele 2 entitati fiind separate.

```
<?php
 // pun tot continutul fisierului intr-o variabila
$continut = file_get_contents( 'fisier.txt' );
 // in acest moment tot continutul este stocat in variabila $continut
// pot procesa continutul sau il pot modifica
 // adaug ceva la final
// Nota: modificarea se face doar in variabila $continut, nu si in fisier
$continut = $continut . ' -- Text adaugat de PHP';
 // la final salvez variabila $continut intr-un al doilea fisier
file_put_contents( 'fisier2.txt', $continut );
 // creea un alt fisier care contine 2 cifre doar
file_put_contents( 'fisier3.txt', '11' );
```

```
// din continutul unui fisier creez un vector format din liniile fisierului
$linii = file( 'fisier4.txt' );
 echo 'Linia a 3a din fisier este: ', $linii[ 2 ];
 ?>
```

2.2 Manipularea fisierelor in PHP 0 !

Pe langa functiile pentru preluarea/modificarea continutului fisierelor, PHP ofera si facilitati de gestionare a fisierelor: creare, mutare, copiere, modificare atribute, etc. Cateva din acestea sunt prezentate mai jos.

```
<?php
 // verificare daca un fisier exista
echo file_exists( 'fisier.txt' );
 // copiere fisier
copy( 'sursa.txt', 'destinatie.txt' );
 // stergere fisier
unlink( 'fisier.txt' );
 // redenumire sau mutare
rename( 'vechi.txt', 'nou.txt' );
 // afisarea numelui unui fisier
echo basename( "c:\\cale\\catre\\fisier.txt" ); // afiseaza fisier.txt
 // afisarea folderului unui fisier
echo dirname( "c:\\cale\\catre\\fisier.txt" ); // afiseaza c:\cale\catre
```

```
// afisarea dimensiunii unui fisier in octeti
echo filesize( 'fisier.txt' );
 // verificare daca numele specificat este un fisier
echo is_file( "c:\\cale\\catre\\fisier.txt" ); // afiseaza true
 // verificare daca un fisier poate fi citit
echo is_readable( 'fisier.txt' );
 // verificare daca un fisier poate fi scris/modificat
echo is_writable( 'fisier.txt' );
 ?>
```

2.3 Directoare in PHP 0 !

Manipularea directoarelor (folderelor) folosind PHP se face la fel de usor ca in cazul fisierelor. Majoritatea functiilor folosite pentru fisiere se pot aplica si la foldere (de exemplu copy, rename, is_file, etc), dar exista o serie de alte instructiuni specifice dosarelor.

```
<?php
 // afiseaza directorul curent (current working directory)
// de obicei este folderul in care se afla scriptul ce se executa
echo getcwd(); // ex. c:\scripturi
 // schimba directorul curent
chdir( 'exemple' ); // ex. c:\scripturi\exemple
 // returneaza un vector ce contine numele fisierelor si directoarelor dintr-un folder
$vector = scandir( getcwd() );
```

```
print_r( $vector );

 // verifica daca un element este director (folder)

echo is_dir( "c:\\cale\\fisier.txt" );    // afiseaza false

 ?>
```

Limbajul PHP dispune de o serie de functii ce permit citire continutului unui folder intr-un mod similar cu preluarea continutului unui fisier. Astfel, exista functii pentru deschiderea unui director (opendir), citirea continutului, adica a fisierelor sau folderelor existente in acel director (readdir) si inchiderea lui (closedir). O situatie in care aceste functii pot fi folosite este aceea cand se doreste afisarea unei liste a elementelor continute intr-un folder si se doreste efectuarea unor calcule sau procesari pe baza fiecaruia dintre aceste elemente.

```
<?php

 $folder = getcwd();            // va lista folderul curent

$handle = opendir( $folder );   // deschid folderul

 if ( !empty( $handle ) ) {

   echo "Fisiere si directoare:\n";

    $terminat = false;

   while ($terminat == false) {

      $file = readdir($handle);   // citesc urmatorul fisier

      if( $file === false ) {

         // atentie la operatorul de exactitate ===

         // daca nu mai sunt alte fisiere/foldere trebuie sa ies din bucla

         $terminat = true;

      } else {

         // aici pot face orice procesare, de exemplu sa redenumesc fisierul/subdirectorul

         // doar afisez numele fisierului/subdirectorului
```

```
            echo "$file\n";
            echo "<br />";
        }
    }

    closedir($handle);   // inchei citirea din folder
}
?>
```