6 HEADERE HTTP, COOKIES SI SESIUNI

6.1 Headere HTTP

6.1.1 Principii

Asa cum, in sistemul de fisiere, pe langa informatia utila dintr-un fisier sunt stocate si informatii *despre* acesta (data crearii, a ultimei modificari etc), la fel intr-un mesaj HTTP sunt transmise, pe langa datele efective, informatii de control (ex: informatii despre continutul mesajului, despre client, server etc). Aceste informatii aditionale poarta denumirea de *headere* HTTP si sunt importante pentru programatorul PHP deoarece prin intermediul lor se realizeaza operatii precum:

- redirectionarea clientului catre o alta adresa (prin intermediul headerului de tip Redirect)
- specificarea tipului de continut pe care serverul il trimite clientului, astfel incat browserul web sa stie cum sa trateze acel continut: sa il afiseze direct (daca este vorba de text simplu), sa il afiseze formatat (daca este sursa HTML), sa il deschida cu ajutorul unui plug-in[1] (daca este de tip PDF sau doc) etc. Toate acestea sunt posibile prin intermediul headerului HTTP Content-type
- setarea unui cookie (memorarea unei informatii pe client, in scopul extragerii ei ulterioare) – cu ajutorul headerului Set-Cookie (vezi sectiunea despre cookies in cadrul acestui material)
- specificarea strategiei de caching[2] a browserului pentru pagina ceruta – cu ajutorul headerelor Cache-Control si Expires. Browserele internet incearca in general sa pastreze in cache cat mai multa informatie posibila, astfel incat utilizatorul sa aiba dublul avantaj al vitezei si al consumului mai mic de banda. Uneori insa pastrarea unei copii a unei pagini pe client nu este dezirabila (ex: pentru paginile generate dinamic, al caror continut se poate schimba de la o cerere la alta)

In mesajul HTTP, headerele se afla intotdeauna inaintea continutului mesajului si sunt separate de acesta printr-o linie goala. De aceea este important ca, atunci cand o aplicatie genereaza dinamic continut web (inclusiv headere HTTP atasate acestui continut), headerele sa fie generate inaintea oricarui alt output al aplicatiei generatoare.

6.1.2 Lucrul cu headere HTTP din PHP

PHP pune la dispozitia programatorului cateva functii predefinite pentru lucrul cu headere HTTP:

- **header()** – folosita pentru a trimite headere HTTP specificate de programator catre client
- **headers_sent()** – folosita pentru a verifica daca headerele au fost deja trimise catre client (caz in care nu mai putem adauga altul)
- **headers_list()** – returneaza un tablou cu lista de headere destinate clientului (care au fost deja trimise sau care sunt in asteptare)

Functia header are urmatorul prototip:

```
void header ( string $header [, bool $replace [, int $http_response_code]] )
```

Iata semnificatiile argumentelor:

- **$header** - reprezinta headerul HTTP in forma in care acesta apare in mesajul HTTP (exemplu: “Location: http://www.example.com/index.php”)

[1] Plugin – un modul care se poate adauga unei aplicatii pentru a-i extinde functionalitatea. In cazul browserului web, plugin-urile ii ofera posibilitatea de a afisa mai multe tipuri de continut
[2] Caching – in cazul browserului web, pastrarea unei copii locale a informatiilor primite de la server in scopul refolosirii lor rapide

- **$replace** – indica daca, in caz ca exista deja un header cu acelasi nume, noul header sa il inlocuiasca pe cel vechi sau sa fie adaugat listei de headere ce trebuie trimise cilentului.
- **$http_response_code** – specifica codul de raspuns HTTP care sa fie trimis clientului. Codurile de raspuns HTTP sunt formate din 3 cifre, iar in functie de prima cifra pot avea urmatoarele semnificatii:
  - **1xx** – coduri de informare a clientului, intermediare
  - **2xx** – coduri ce indica succesul unei cereri a clientului
  - **3xx** – redirectionari
  - **4xx** – indica o eroare din partea clientului (ex: 404 – Not Found)
  - **5xx** – erori ale serverului sau incapacitatea acestuia de a onora cererea clientului (ex: 503 Service Unavailable)

Exemple:

```
header('Content-type: application/pdf');
// urmeaza a se trimite clientului continut de tip PDF, generat sau citit dintr-un fisier
```

```
// redirectionarea clientului catre o alta adresa
header("Location: http://www.example.com");
exit;
```

Nota: *unele headere HTTP setate cu functia header() au ca efect secundar trimiterea unui anume cod HTTP catre client. Exemplu: header("Location: http://www.infoacademy.net"); va determina trimiterea codului 302 (Redirect)*

```
// trimiterea unui fisier .doc prin intermediul unui script PHP
header("Content-type: text/html");

// fortam aparitia ferestrei de save in loc de simpla afisare formatata in browser
header("Content-disposition: attachment; filename=statistici.html");

// transmiterea continutului fisierului
readfile("c:/stats.html");
```

Atentie! Nu uitati ca headerele trebuie trimise inaintea oricarui alt output! Un simplu spatiu sau o linie goala generate de catre scriptul PHP inseamna deja inceputul output-ului, iar dupa aceea incercarea de trimitere de headere se va solda cu o eroare.

```
if(!headers_sent())
      header("Cache-Control: no-cache, must-revalidate");
```

6.2 Memorarea de informatii intre doua cereri HTTP

6.2.1 Caracterul *stateless* al protocolului HTTP (de ce este nevoie de memorarea starii?)

HTTP este un protocol *stateless* – fiecare cerere este tratata de catre server independent de celelalte, serverul nu isi "aminteste" ce s-a intamplat la cererile anterioare si nu coreleaza in vreun fel mai multe cereri, chiar daca ele provin de la acelasi client sau daca corespund unor resurse aflate in acelasi site.

Exista numeroase scenarii in care am dori sa propagam informatii de la o cerere a clientului la alta. Iata doua, clasice:

- un site in care clientii, odata autentificati, au acces la o serie de pagini interne (protejate). Aceasta presupune doua etape:
  - serverul autentifica clientul, ca urmare a unei cereri a clientului in care acesta transmite combinatia user/parola corecta (in cazul in care autentificarea se face pe baza de user/parola). Serverul transmite clientului o informatie ce va permite identificarea lui ulterioara
  - serverul recunoaste celelalte cereri ale clientului ca venind de la acelasi client, care figureaza ca deja autentificat, si in consecinta ii va permite acestuia accesul la paginile protejate
- un site de comert online, unde utilizatorul navigheaza prin diferite categorii de produse si le adauga in cos pe cele dorite. Aceasta presupune ca serverul sa recunoasca toate cererile clientului ca venind de la acelasi utilizator si, in plus, sa pastreze continutul cosului de cumparaturi de la o cerere la alta

In rezumat, ceea ce isi doreste programatorul web este ca serverul sa poata mentine informatii despre un client (daca este sau nu autentificat, username-ul/id-ul sau, continutul cosului de cumparaturi etc) de-a lungul unei succesiuni de cereri pe care acesta le efectueaza catre server – cu alte cuvinte, posibilitatea de a delimita *sesiuni* ale utilizatorului in interactiunea sa cu aplicatia care ruleaza pe server.

6.2.2 Solutii pentru memorarea informatiei intre doua cereri

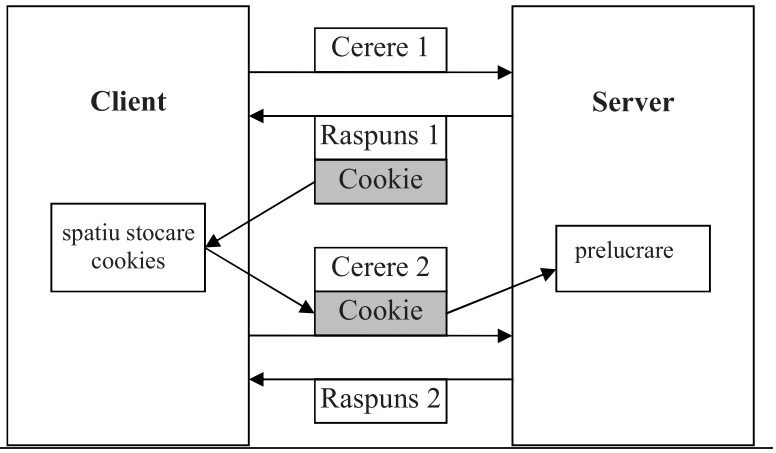
Pentru ca serverul sa poata mentine informatii care tin de un anume client de la o cerere a acestuia la alta, este necesara salvarea datelor in cauza la incheierea unei cereri si incarcarea lor la cererea urmatoare. Memorarea acestor date se poate face:

- **pe client**. Solutia este mecanismul de cookies – mici cantitati de informatie pe care serverul web le poate stoca pe hard-disk-ul clientului, urmand ca ele sa-i fie trimise inapoi serverului la cererile urmatoare pe care clientul le efectueaza (vezi mai jos sectiunea despre cookies)
- **pe server**. Solutia este sistemul de management al sesiunilor: serverul aloca clientului un spatiu de stocare pentru informatii proprii (variabile de sesiune), in care acestea se salveaza la incheierea fiecarei cereri si se incarca inapoi la urmatoarea cerere care este detectata a veni de la acelasi client. Mecanismul de stocare este dublat de unul de identificare a clientului, care permite serverului sa recunoasca cereri disparate ca venind de la acelasi client (vezi sectiunea despre sesiuni). Felul in care se face salvarea informatiilor de sesiune este configurabil de catre administratorul serverului web si al modulului de PHP (se poate face in fisiere, baze de date etc.).

6.3 Cookies

6.3.1 Principii generale

Cookie-urile reprezinta perechi nume-valoare pe care serverul web le poate stoca pe hard-disk-ul clientului, prin intermediul browserului web. Ele sunt transmise clientului prin intermediul unui header HTTP (Set-Cookie sau Set-Cookie2), in cadrul unui raspuns la o cerere a clientului. La urmatoarele cereri depuse de acelasi client catre acelasi server, clientul va trimite automat continutul cookie-ului catre



server, care poate folosi datele respective ca si date de intrare ale unei aplicatii ce genereaza o pagina web (asemanator cu informatiile provenite prin GET sau POST). Astfel, cookie-urile sunt un mecanism de propagare a informatiei de la o cerere la alta.

Iata doua utilizari tipice pentru cookie-uri:

- un site care contine o sectiune restrictionata, accesibila numai utilizatorilor autentificati, seteaza un cookie in browserul clientului in momentul in care acesta se autentifica. Cookie-ul contine id-ul clientului si va fi trimis automat serverului la cererile urmatoare efectuate de client, ceea ce va permite serverului sa-l identifice si sa-i trateze cererile ca parte a unei sesiuni
- un site care permite utilizatorului sa specifice preferinte (ex: limba dorita, culoarea etc); odata alese, aceste preferinte sunt stocate pe hard-disk-ul clientului intr-un cookie si sunt trimise serverului la fiecare cerere ulterioara, ceea ce ii da acestuia posibilitatea de a genera de fiecare data pagina in concordanta cu preferintele exprimate anterior de client

Lucrul cu cookie-uri prezinta urmatoarele caracteristici:

- avantaje
  - un cookie poate memora informatie in mod persistent, intre doua cereri ale clientului, chiar daca ele sunt efectuate la momente de timp indepartate (ex: zile)
  - mecanismul de cookies functioneaza in mod transparent pentru utilizator – browserul si serverul web fac operatiile necesare pentru mentinerea starii intre cereri
- dezavantaje
  - utilizatorul poate sterge cookie-urile din browser
  - browserul poate fi configurat sa nu accepte cookies sau sa le accepte selectiv
  - browserele ofera de obicei spatiu de stocare drastic limitat pentru cookies

6.3.2 Parametrii unui cookie

Fiecare cookie trimis clientului are urmatorul set de parametri:

- **nume** – cel prin intermediul caruia vor fi accesate datele de catre server atunci cand cookie-ul este trimis inapoi de catre client
- **valoare** – informatiile stocate in cookie
- **data de expirare** – specifica momentul stergerii cookie-ului; daca lipseste, cookie-ul este sters atunci cand userul inchide browserul. Serverul poate schimba acest parametru pentru un cookie deja setat (de exemplu, poate forta stergerea lui setand data de expirare in trecut)
- **domeniu** si **cale** – specifica in ce conditii este trimis inapoi cookie-ul catre server: numai daca domeniul este acelasi si URI-ul cerut de client se afla in interiorul caii specificate in *cale*
- **secure** – pune conditia ca cookie-ul sa nu fie transmis serverului decat daca conexiunea acestuia cu clientul este una criptata (HTTPS)
- **http only** – pune conditia ca cookie-ul sa fie accesibil doar serverului web prin intermediul protocolului HTTP, nu si limbajelor de scripting locale (ex: Javascript)

Parametrii **nume** si **valoare** sunt obligatorii, toti ceilalti sunt optionali.

6.3.3 Lucrul cu cookies din PHP

6.3.3.1 Setare cookie

Numim setarea unui cookie operatia prin care serverul trimite un cookie catre client prin intermediul headerului HTTP corespunzator, determinand fie crearea cookie-ului in browserul clientului, fie modificarea parametrilor unui cookie setat anterior.

Functia PHP predefinita folosita in acest scop este setcookie:

```
setcookie ($nume [,$valoare [, $expirare [, $cale [, $domeniu [, $secure [, $httponly]]]]]])
```

Functia returneaza FALSE daca scriptul a produs deja output (si deci headerul HTTP Set-Cookie nu mai poate fi trimis).

Argumentele functiei se mapeaza pe parametrii unui cookie, cu urmatoarele mentiuni:

- valoarea unui cookie nu poate fi decat scalar
- momentul expirarii
    - se specifica sub forma de Unix time (numarul de secunde scurs de la 1 ianuarie 1970, care este data conventionala a crearii sistemului de operare Unix). Functia PHP **time()** intoarce momentul prezent exprimat in acest fel, ceea ce o face foarte potrivita pentru setarea momentului expirarii unui cookie
    - daca nu este specificat sau setat pe valoarea 0, cookie-ul va expira cand utilizatorul inchide browserul

Exemple:

```
// setare cookie doar cu nume si valoare; expira la inchiderea browserului
setcookie("c_id", 4167);

// setare cookie cu specificarea momentului expirarii: 2 minute de la setare
setcookie("color","red",time()+60);
```

Atentie! *Deoarece trimiterea cookie-urilor se face prin intermediul unui header HTTP, toate cookie-urile trebuie setate inaintea oricarui output al scriptului PHP!*

Daca se doreste trimiterea intr-un cookie (si recuperarea ulterioara) a mai multor valori, exista doua posibilitati:

- se seteaza mai multe cookie-uri. Daca se doreste ca, la citirea valorii acestora, sa se obtina un tablou (vezi mai jos accesarea informatiei din cookies), numele cookie-urilor pot fi setate cu sintaxa de tablou:

```
setcookie("cookie[0]", 3);
setcookie("cookie[1]", 5);
```

- se includ valorile in cauza intr-un tablou, care este transformat intr-un string folosind functiile serialize() sau implode()

6.3.3.2 Accesarea informatiei provenite din cookies

Clientul trimite automat impreuna cu cererea HTTP toate acele cookie-uri ale caror parametri *domeniu* si *cale* corespund cu URL-ul solicitat in cerere. Daca URL-ul determina executia unui script PHP, interpretorul PHP primeste de la server informatia din cookie si o stocheaza in variabila $_COOKIE – un tablou asociativ de tip superglobal. Acest tablou contine cate un element pentru fiecare cookie primit, perechea cheie-valoare fiind reprezentata de numele si valoarea cookie-ului.

Atentie! *Un cookie este setat intr-un raspuns al serverului si este disponibil acestuia abia incepand cu cererea urmatoare!*

```
if(isset($_COOKIE['lang'])){
      $preferred_language = $_COOKIE['lang'];
else{
      $preferred_language = "en";
      setcookie("lang","en");
}
```

6.3.3.3 Stergerea unui cookie

Stergerea unui cookie se poate face in diferite moduri:

- ca urmare a unei actiuni a utilizatorului: browserele din ziua de astazi ofera utilizatorului interfete facile pentru managementul cookie-urilor, inclusiv stergerea sau modificarea parametrilor acestora
- ca urmare a unei actiuni efectuate pe server:
    - programatorul poate da valoarea vida cookie-ului (trimitand un cookie cu parametri identici dar avand ca valoare sirul vid)
    - programatorul poate seta momentul expirarii cookie-ului in trecut

Exemplu: pentru un cookie setat initial astfel:

```
// setarea initiala a cookie-ului
setcookie("nume","Mihai",time()+2000);
```

modalitatile de stergere sunt:

```
// … la o cerere ulterioara, cookie-ul este suprascris, dandu-i-se valoarea "" :
setcookie("nume", "");

// …alternativ, se putea muta in trecut momentul expirarii
setcookie("nume", "Mihai", time()-86400);
```

Nota: *este o practica buna ca, la stergerea unui cookie, timpul de expirare sa fie setat mult in trecut, deoarece ora de pe server si cea de pe client pot diferi semnificativ, iar decizia stergerii cookie-ului la expirare ii apartine browserului, care ruleaza pe client.*

6.4 Sesiuni

6.4.1 Principii generale

O sesiune reprezinta o succesiune de cereri ale unui client web care sunt recunoscute de catre server ca venind de la acelasi utilizator. Trebuie inteles ca, desi pentru client experienta este una cursiva (succesiune de pagini accesate), pentru server acestea sunt niste cereri disparate, aflate la distante variabile in timp si intretesute cu cereri ale altor clienti ce efectueaza cereri catre acelasi server.

Observatie: *o sesiune presupune ca serverul sa isi dea seama ca mai multe cereri vin de la acelasi* utilizator*, nu de la aceeasi adresa IP. In spatele unei adrese IP se poate afla o statie pe care sunt deschise doua aplicatii browser diferite, sau o intreaga retea care foloseste adresa in cauza pentru a avea acces la internet.*

O sesiune poate dura:

- din momentul in care clientul se conecteaza la server pana cand inchide browserul. Exemplu: un site care incearca sa-i afiseze utilizatorului de fiecare data alta reclama. Pentru aceasta, serverul trebuie sa tina evidenta reclamelor deja afisate si, in plus, sa recunoasca toate cererile clientului ca facand parte din aceeasi sesiune de browsing
- dintr-un moment ulterior primei cereri efectuate de client, pana intr-un alt moment, anterior inchiderii browserului. Exemplu: un student InfoAcademy intra pe site-ul www.infoacademy.net, citeste descrierea catorva cursuri pe care intentioneaza sa le urmeze in viitor si apoi isi introduce user/pass pentru a accesa zona restrictionata a site-ului, in scopul sustinerii de examene. Cererile efectuate de client anterior login-ului pot fi tratate independent, insa, odata ce utilizatorul s-a logat, toate cererile provenite de la el trebuie recunoscute ca venind de la acelasi utilizator; calitatea sa de client autentificat va face serverul sa-i prezinte pagini la care in mod normal nu ar fi avut acces. Odata ce studentul face logout, sesiunea se inchide, iar cererile clientului nu vor mai primi tratament special, ci vor fi onorate in mod independent.

Cat timp clientul continua sa efectueze cereri, sesiunea se mentine activa. O sesiune se incheie atunci cand utilizatorul a terminat de interactionat cu paginile site-ului sau portiunii de site ce fac obiectul sesiunii. Serverul nu poate identifica intotdeauna clar momentul de timp in care sesiunea s-a incheiat din punct de vedere al utilizatorului: un interval mare de inactivitate din partea acestuia ar putea indica faptul ca utilizatorul zaboveste mai mult timp asupra unei pagini, sau ca a inchis fereastra de browser. Din acest motiv, pe server se poate configura un timp maxim cat serverul mentine sesiunea activa in lipsa unei noi cereri de la client; dupa ce se scurge acest interval de timp, are loc inchiderea automata a sesiunii (session time out).

6.4.2 Caracteristici ale sistemului de sesiuni in PHP

6.4.2.1 Generalitati

In PHP, managementul sesiunilor pe server implica doua aspecte:

- identificarea in mod unic a utilizatorului, astfel incat cererile sale sa poata fi considerate ca facand parte din aceeasi sesiune
- memorarea informatiilor ce tin de sesiunea utilizatorului in cauza intre doua cereri succesive ale acestuia

6.4.2.2 Identificarea corespondentei utilizator⇔sesiune

Fiecare sesiune PHP este identificata printr-un session ID – un identificator unic, generat de catre interpretor sau setat de catre programator. Acesta trebuie trimis de catre client serverului la fiecare cerere, astfel incat serverul sa poata incadra in aceeasi sesiune cererile clientului si sa-i ofere acces la informatiile de sesiune. Propagarea session ID-ului de la o cerere la alta se poate realiza in doua feluri:

- **printr-un cookie stocat pe client**. Odata cookie-ul setat, clientul va trimite serverului cookie-ul catre server la fiecare cerere ulterioara, si astfel serverul va sti datele carei sesiuni sa le faca disponibile acelui client. Aceasta este modalitatea default de propagare a session ID-ului in PHP
- **in cadrul URL-urilor pe care le acceseaza clientul** si care trimit catre acelasi server. Aceasta presupune ca fiecare cerere a clientului (fie ea GET sau POST) sa contina session id-ul clientului, care sa poata fi preluat de catre server si folosit pentru a identifica sesiunea dorita.

Exemplu: propagarea session id-ului in URL:

```
http://www.example.com/index.php?PHPSESSID=a14bee3fc818279c9d6129a9c0a612f1
```

6.4.2.3 Memorarea informatiilor de sesiune

Informatiile de sesiune in PHP sunt variabile pe care programatorul le defineste si care se propaga de-a lungul a doua sau mai multe cereri ale aceluiasi client.

Propagarea informatiilor de sesiune (session data) presupune:

- salvarea informatiilor la incheierea unei cereri a clientului
- incarcarea lor la urmatoarea cerere pe care serverul o recunoaste ca venind de la acelasi client

Variabilele de sesiune stau in tabloul asociativ de tip superglobal $_SESSION. Acest tablou este populat automat cu informatiile deja salvate atunci cand interpretorul primeste de la client un session id si acestui id ii corespunde o sesiune deja existenta. De asemenea, orice element nou adaugat in $_SESSION va fi salvat impreuna cu celelalte odata ce executia scriptului PHP se incheie.

Modulul PHP pentru lucrul cu sesiuni salveaza toate variabilele unei sesiuni intr-un fisier la incheierea cererii si le incarca inapoi la urmatoarele cereri ale clientului. Informatiile de sesiune sunt practic restaurate la fiecare cerere a aceluiasi client, indiferent daca diversele cereri se refera la acelasi script PHP sau la scripturi diferite. La fiecare salvare a sesiunii, este schimbata data de modificare a fisierului, astfel incat serverul sa stie cand trebuie sa stearga sesiunea in caz de timeout.

Nota: *locatia de pe hard-disk unde se salveaza fisierele cu datele de sesiune se stabileste cu parametrul de configurare* ***session.save_path*** *din fisierul php.ini sau cu ajutorul functiei* ***session_save_path()*** *(efectul acesteia din urma durand numai pe parcursul executiei scriptului)*

Observatie: *fisierul sesiune nu se creeaza decat daca $_SESSION contine cel putin un element (daca $_SESSION este gol, practic nu exista date de sesiune si nu are rost crearea fisierului pentru ca nu avem ce restaura)*

6.4.3 Lucrul cu sesiuni in PHP

6.4.3.1 Crearea sau restaurarea unei sesiuni; stabilirea parametrilor

PHP ofera programatorului functia **session_start()**, care poate avea efecte diferite in functie de caz:

- daca interpretorul a primit de la client un session id, este cautat fisierul sesiune corespunzator:
  - daca este gasit, sunt restaurate datele din $_SESSION
  - daca nu este gasit, se creeaza o sesiune noua cu id-ul specificat
- daca interpretorul nu se afla in posesia session id-ului, este generat un nou session id si se creeaza o sesiune noua

In ambele cazuri, functia session_start() trimite catre client asa-numitul *session cookie* – un cookie ce contine ID-ul sesiunii, si pe care clientul il va trimite inapoi serverului la cererile urmatoare.

Atentie! *Deoarece functia* ***session_start()*** *are ca efect trimiterea catre client a headerului HTTP Set-Cookie, ea trebuie apelata inaintea oricarui output al scriptului!*

Parametrii sesiunii sau ai session cookie-ului se pot obtine/seta cu alte cateva functii PHP, inaintea apelarii lui session_start():

- **session_name**([$nume]) – returneaza numele session cookie-ului. Apelata cu argument, il seteaza.
- **session_id**([$id]) – idem pentru session ID
- **session_set_cookie_params**(int $lifetime [, string $path [, string $domain [, bool $secure [, bool $httponly]]]]) – stabileste ceilalti parametri ai session cookie-ului (in afara de nume si valoare)
- **session_get_cookie_params()** – returneaza parametrii session cookie-ului sub forma de tablou asociativ

Nota: *parametrii session cookie-ului au valori default stabilite in fisierul php.ini, cu ajutorul directivelor din seria session.cookie_*. Numele din oficiu al session cookie-ului este PHPSESSID, iar expirarea cookie-ului are loc la inchiderea browserului*

Exemplu: un script ce afiseaza initial un formular care solicita numele utilizatorului, si care il saluta apoi in mod diferit dupa introducerea numelui sau la o revenire ulterioara:

```
<?php

// crearea unei sesiuni noi sau restaurarea uneia existente, dupa caz
session_start();
// ca urmare, s-a trimis catre client un cookie cu numele PHPSESSID (daca
// setarile default din php.ini au fost pastrate)

// daca sesiunea este una restaurata si contine numele utilizatorului, il salutam
if(isset($_SESSION['nume'])){
      echo "Salut $_SESSION[nume], bine ai revenit!";
}else{
      // daca am primit numele prin POST, il memoram in sesiune
      if(isset($_POST['nume'])){
            $_SESSION['nume'] = $_REQUEST['nume'];
            echo "Bine ai venit pe site-ul nostru, $_SESSION[nume]!";
      }else
            // nu avem nume, i-l solicitam utilizatorului prin intermediul formularului
            echo <<<FORM
```

```
            <form method="post" action="$_SERVER[PHP_SELF]">
            Nume: <input type="text" name="nume">
            <input type=submit value="Trimite">
            </form>
FORM;
}
?>
```

La prima apelare a acestui script, clientul nu a furnizat inca nume, de aceea ii este afisat formularul. Daca utilizatorul introduce numele Mihai si apasa "Trimite", i se va afisa mesajul "Bine ai venit pe site-ul nostru, Mhai!". La orice cerere ulterioara va fi afisat mesajul "Salut Mihai, bine ai revenit!".

De remarcat faptul ca, odata salvata o variabila de sesiune, ea este disponibila si altor scripturi, nu numai celui in care a fost creata. Sa presupunem ca pe acelasi server cu scriptul anterior se afla inca unul, care afiseaza suma pozitiilor alfabetice ale literelor numelui in vederea analizei numerologice:

```
<?php
session_start();
// daca exista o sesiune salvata anterior, $_SESSION['nume'] va primi valoarea salvata

// facem calcule numai daca, dupa restaurarea sesiunii, avem numele persoanei
if(!empty($_SESSION['nume'])){
      // convertim numele la litere mici
      $nume = strtolower($_SESSION['nume']);
      $suma = 0;
      for($i=0; $i<strlen($nume); $i++){
            /* caracterul "a" are pozitia alfabetica 1; scazand din codul caracterului
            curent codul lui "a" si adunand 1 rezulta pozitia alfabetica a caracterului
            curent*/
            $pozitie_alfabetica = ord($nume[$i])-ord("a")+1;
            $suma += $pozitie_alfabetica;
      }
      echo "Draga $_SESSION[nume], numarul corespunzator numelui tau este $suma";
}
?>
```

Daca utilizatorul acceseaza acest script (de exemplu, ca urmare a click-ului pe un link) dupa introducerea numelui "Mihai" in cel precedent, ii va fi afisat direct mesajul "Draga Mihai, numarul corespunzator numelui tau este 40".

6.4.3.2 Inchiderea unei sesiuni

O sesiune se poate inchide in diferite situatii:

- daca session cookie-ul este pastrat cu timpul de expirare default, el va fi sters cand utilizatorul inchide browserul si, implicit, utilizatorul nu va mai putea accesa sesiunea, aceasta urmand sa fie stearsa de pe server dupa timeout
- daca clientul nu efectueaza nici o cerere un timp mai lung decat timeout-ul sesiunii, sesiunea este stearsa de pe server
- programatorul poate inchide sesiunea explicit, efectuand urmatorul set de operatii:
    - golirea tabloului $_SESSION
    - apelarea functiei **session_destroy**(), care sterge datele corespunzatoare sesiunii
    - stergerea session cookie-ului

```php
<?php
session_start();

// golirea tabloului $_SESSION
$_SESSION=array();

// schimbare parametri session cookie: valoare vida si timp de expirare in trecut
setcookie(session_name(), "", time()-1000);
// operatia de mai sus are sens doar daca session id-ul este propagat folosind cookies!

// stergerea fisierului sesiune
session_destroy();
?>
```

6.5 BIBLIOGRAFIE

- The Unofficial Cookie FAQ: http://www.cookiecentral.com/faq/
- PHP Manual – Session Handling Functions: http://ro.php.net/manual/en/intro.session.php
- Passing the session id: http://ro.php.net/manual/en/session.idpassing.php
- PHP 5 and MySQL Bible (O'Reilly): Sessions, Cookies and HTTP (pag 455 – 478)
- Zend PHP 5 Certification Study Guide – HTTP Headers, Sessions (pag 104-110)
- Cookies and Sessions: http://hudzilla.org/phpwiki/index.php?title=Cookies_and_sessions