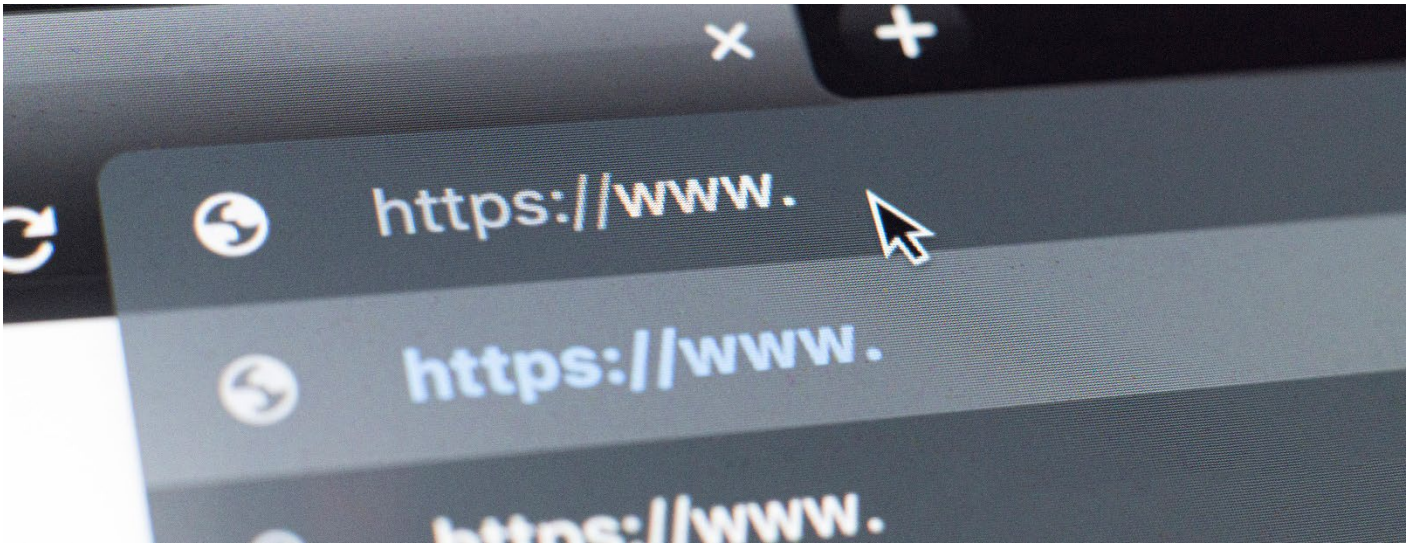
CAP. 8
SCHIMBUL DE DATE CLIENT-SERVER

8.1 Protocolul HTTP

Hypertext Transfer Protocol (**DNS**) este un protocol din stiva TCP/IP ce lucrează la nivelul aplicație. Este metoda cea mai des utilizată pentru accesarea informațiilor din Internet care sunt păstrate pe servere World Wide Web (WWW). Protocolul HTTP este un protocol de tip text, fiind protocolul "implicit" al WWW. Din 1990 HTTP este un protocol generic și apatrid care poate fi utilizat și în alte scopuri extinzând metodele de cerere, codurile sale de eroare sau antetele. Practic, HTTP este un protocol de comunicare, care este folosit pentru a furniza date (fișiere HTML, fișiere imagine, rezultate ale interogării, etc.) de pe World Wide Web. Acesta oferă un mod standardizat pentru prin care calculatoarele comunică unul cu altul.

HTTP oferă o tehnică de comunicare prin care paginile web se pot transmite de la un server aflat la distanță pe calculatoarele (browser-ele) clienților. Dacă accesează un link sau o adresă web cum ar fi **http://www.example.com**, atunci clientul cere (indirect) calculatorului său să afișeze o pagină web (de obicei **index.html**) aflată pe server. În prima fază numele (adresa) **www.example.com** este convertit de protocolul Domain Name System (**DNS**) într-o adresă IP. Transferul datelor se face prin protocolul Transmission Control Protocol (**TCP**) pe portul standard **80** al serverului HTTP, ca răspuns la cererea HTTP formulată de client. Informații suplimentare precum indicații pentru browser, limba dorită ș.a. se pot adăuga în header-ul (antetul) pachetului HTTP.

În urma cererii (realizată de multe ori cu metoda GET a protocolului HTTP) venită din partea clientului, serverului furnizează răspunsul sub forma datelor cerute. Acestea vin sub formă de pagini (X)HTML, fișiere atașate (imagini), fișiere de stilizare (CSS), scripturi (Javascript), dar pot fi și pagini generate dinamic (SSI, JSP, PHP și ASP.NET). Dacă dintr-un anumit motiv informațiile nu pot fi transmise, atunci serverul trimite înapoi un mesaj de eroare. Modul exact de desfășurare a acestei acțiuni (cerere și răspuns) este stabilit în specificațiile HTTP.



8.1.1 Coduri de răspuns HTTP

Codurile primite ca răspuns în cadrul protocolului HTTP sunt clasificate în 5 clase (categorii). Cele mai utilizate coduri sunt prezentate în tabelul următor:

Grup de coduri	Semnificație	Exemple
1XX	Informare	100 – serverul acceptă tratarea cererii de la client 101 – schimbare de protocol
2XX	Succes	200 – cerere reușită 202 – cerere acceptată 204 – nu există conținut
3XX	Redirecționare	301 – pagină mutată definitiv 302 – pagină mutată temporar 304 – pagina din memoria ascunsă este încă validă
4XX	Eroare client	401 – cererea necesită autentificare 403 – pagină interzisă 404 – pagina nu a fost găsită
5XX	Eroare server	500 – eroare internă la server 503 – încearcă mai târziu

8.1.2 Metode HTTP

Metodele disponibile în cadrul protocolului HTTP pentru interacțiunea cu serverul web sunt prezentate în cadrul tabelului următor:

Metodă	Descriere
GET	este cea mai folosită metodă, fiind utilizată atunci când serverului i se cere o resursă
HEAD	se comportă exact ca metoda GET, dar serverul returnează doar antetul resursei, ceea ce permite clientului să inspecteze antetul resursei, fără a fi nevoit să obțină și corpul resursei
PUT	metoda este folosită pentru a depune documente pe server, fiind inversul metodei GET
POST	a fost proiectată pentru a trimite date de intrare către server
DELETE	este opusul metodei PUT
TRACE	este o metodă folosită de obicei pentru diagnosticare, putând da mai multe informații despre traseul urmat de legătura HTTP
OPTIONS	este folosită pentru identificarea capacităților serverului Web, înainte de a face o cerere
CONNECT	este o metodă folosită în general de serverele intermediare

8.1.3 Metoda GET

Această metodă este responsabilă pentru descărcărea resursei specificate de pe serverul web pe client. Majoritatea cererilor către un server web sunt de acest tip.

Când utilizatorul completează un formular ca cel de mai jos și apasă butonul de trimitere (submit), datele formularului sunt trimise pentru procesare într-un fișier PHP aflat pe server.

```
<form action="welcome.php" method="get">
    Nume: <input type="text" name="name"><br>
    Varsta: <input type="text" name="age"><br>
    <input type="submit">
</form>
```

În exemplul de mai jos, codul este plasat într-un fișier denumit „**welcome.php**”. Datele formularului anterior sunt trimise cu metoda HTTP **GET**, iar variabila super globală PHP $_GET este folosită pentru a accesa datele transmise din formular:

```
<?php
if ($_GET["name"] || $_GET["age"]) {
    echo "Bine ai venit " . $_GET["name"] . "<br>";
    echo "Tu ai: " + $_GET["age"] . "ani.";
}
?>
```

Caracteristicile metodei GET sunt următoarele:

- Metoda GET produce un șir lung ce este stocat în browser sau pe server;
- Metoda GET este restricționată la un număr de doar 1024 caractere;
- Niciodată nu trebuie folosită metoda GET pentru a prelua parola sau alte informații sensibile;
- GET nu poate fi folosit pentru a trimite date binare, precum imagini sau documente, către server;
- Datele trimise prin metoda GET pot fi accesate folosind variabila $_GET, ce reprezintă un vector asociativ.

8.1.4 Metoda POST

Metoda POST este preferată pentru transferul de informații de la client către serverul web. În cazul metodei POST partea de formular este identică, singura diferență fiind faptul că metoda utilizată este **post** în loc de **get**.

```
<form action="welcome.php" method="post">
    Nume: <input type="text" name="name"><br>
    Varsta: <input type="text" name="age"><br>
    <input type="submit">
</form>
```

Datele formularului sunt trimise cu metoda HTTP **POST** către același script PHP. Variabila $_POST este folosită pentru a accesa datele transmise din formular.

```php
<?php
if ($_POST["name"] || $_POST["age"]) {
    echo "Bine ai venit " . $_POST["name"] . "<br>";
    echo "Tu ai: " + $_POST["age"] . "ani.";
}
?>
```

Caracteristicile metodei POST sunt următoarele:

- Metoda POST transferă informații prin anteturile HTTP. Informația este "decodată" asemănător cu metoda GET și stocată în antetul cunoscut ca și QUERY_STRING;
- Metoda POST poate fi folosită pentru trimiterea de date binare;
- Datele trimise prin POST trec prin protocolul HTTP ca și antet. Prin folosirea securizată a lui HTTP putem fi siguri că nu avem breșe de securitate;
- PHP oferă accesul la date prin folosirea variabilei globale $_POST.

8.1.5 Variabila $_REQUEST

PHP folosește variabila globală **$_REQUEST** pentru a obține atât conținutul variabilei **$_GET** cât și **$_POST** dar și **$_COOKIE**. Vom prezenta mai multe detalii despre variabila **$_COOKIE** în următorul subcapitol al acestui curs.

```php
<?php
if ($_REQUEST["name"] || $_REQUEST["age"]) {
    echo "Bine ai venit " . $_REQUEST["name"] . "<br>";
    echo "Tu ai: " + $_REQUEST["age"] . "ani.";
}
?>
```

În cazul în care scriptul care procesează cererea se găsește în același fișier cu formularul html este posibil să găsiți în locul atributului **action** următoarea structură:

```php
<form action="<?= $_SERVER["PHP_SELF"] ?>" method="post">
    Nume: <input type="text" name="name"><br>
    Varsta: <input type="text" name="age"><br>
    <input type="submit">
</form>
```

Observație: Variabila `$_SERVER["PHP_SELF"]` conține numele scriptului ce a fost apelat, iar construcția <?= variabilă ?> este echivalentă cu: <?php **echo** variabilă ?>.

Observație: Când inserați orice variabilă în HTML, cu excepția cazului în care doriți ca browser-ul să interpreteze variabila în sine ca și cod HTML, cel mai bine este să utilizați funcția **htmlspecialchars()** împreună cu aceasta. Printre altele, aceasta împiedică hackerii să introducă HTML arbitrar în pagină.

```php
<?= htmlspecialchars($_SERVER["PHP_SELF"]) ?>
```

8.2 Cookies

Cookies sunt fișiere de tip text ce stochează pe computerul clientului date necesare pentru îmbunătățirea și/sau monitorizarea activității utilizatorilor în cadrul paginilor web. PHP suportă cookies transmise prin HTTP.

Ciclul de viață al unui cookie cuprinde următorii 3 pași :

- Aplicația de la nivelul serverului stabilește ce cookies urmează a fi setate în browser. De exemplu, nume, vârsta sau identificator numeric pentru sesiune;
- Browser-ul stochează informația pe computerul clientului;
- Când browser-ul trimite o nouă cerere către același server web atunci el va trimite și aceste informații din cookie-uri iar serverul le va folosi pentru a identifica utilizatorul sau pentru a optimiza conținutul livrat acestuia.

8.2.1 Anatomia unui Cookie

În cele mai multe cazuri cookie-urile sunt folosite în antetul HTTP (cu ajutorul Javascript se poate seta un cookie direct în browser). Un script PHP ce setează un cookie ce ar putea genera următorul rezultat:

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
                path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

După cum se poate observa, setarea se face folosind Set-Cookie având ca și valoare o pereche (nume: valoare), o data GMT, o cale și un domeniu. Numele și valoarea vor fi codate prin URL. Câmpul de expirare este folosit pentru ca browser-ul să șteargă cookie-ul după intervalul de timp specificat.

Observație: Un script PHP va avea acces la cookie prin variabila **$_COOKIE** care păstrează toate numele de cookie-uri și valori asociate.

8.2.2 Setarea Cookie-urilor folosind PHP

PHP oferă funcția **setcookie()** pentru a seta un cookie. Această funcție necesită 6 argumente și trebuie sa fie apelată înainte de eticheta <html>. Pentru fiecare cookie această funcție trebuie să fie apelată separat. Structura funcției este următoarea:

setcookie(*nume, valoare, expirare, cale, domeniu, securizare*);

Observație: Dintre toți acești parametri doar numele este obligatoriu, restul parametrilor sunt opționali.

Parametrii funcției setcookie au următoarele caracteristici:

- **Nume** – Acesta stabilește numele cookie-ului ce va putea fi accesat ulterior prin variabila numită $_COOKIE[‘nume’];
- **Valoare** – Acesta setează valoarea cookie-ului;
- **Expirare** – Acesta specifică un moment din viitor pentru expirarea cookie-ului. Dacă acest parametru nu este setat atunci cookie-ul va expira automat când browser-ul este închis;
- **Cale** – Acesta specifică directorul pentru care cookie-ul este valid. Un singur caracter (“ / ” slash) permite cookie-ului să fie valid în toate directoarele aplicației web;
- **Domeniu** – Acesta poate fi folosit pentru a specifica numele domeniului. Toate cookie-urile sunt valide pentru domeniul găzduit și domeniul pentru care a fost creat.
- **Securizare** – Dacă acest parametru este setat cu valoarea 1 cookie-ul trebuie să fie trimis doar securizat prin folosirea protocolului HTTPS, altfel dacă va fi setat 0, înseamnă ca cookie-ul poate fi trimis prin protocolul standard HTTP.

Următorul exemplu crează 2 cookie-uri, nume și vârstă, ce vor expira după o oră.

```
<?php
setcookie("name", "John Watkin", time() + 3600, "/", "", 0);
setcookie("age", "36", time() + 3600, "/", "", 0);
?>
<html>
<body>
    <h1>Pagina Principala</h1>
</body>
</html>
```

8.2.3 Accesarea Cookie-urilor

PHP oferă mai multe căi de a accesa cookie-urile. Ceea mai simplă cale este de a folosi variabila **$_COOKIE**. Următorul exemplu va accesa cookie-urile stocate în exemplul anterior:

```
<html>
<head>
    <title>Accesare cookies folosind PHP</title>
</head>
<body>
    <?php
        echo "Numele meu este " . $_COOKIE["name"] . "!<br>";
        echo "Varsta mea este de " . $_COOKIE["age"] . " ani";
    ?>
</body>
</html>
```

Putem folosi funcția **isset()** pentru a verifica dacă un cookie este setat sau nu.

```
<html>
<head>
    <title>Accesare cookies folosind PHP</title>
</head>
<body>
```

```
    <?php
    if (isset($_COOKIE["name"])) {
        echo "Bine ai venit " . $_COOKIE["name"] . "!<br>";
    } else {
        echo "Imi pare rau, dar nu te cunosc!";
    }
    ?>
</body>
</html>
```

8.2.4 Ștergerea unui Cookie folosind PHP

Pentru a șterge un cookie, trebuie să apelăm funcția **setcookie()** cu numele cookie-ului dar cu o dată / un timp de expirare din trecut.

```
<?php
    setcookie("name", "", time() - 60);
    setcookie("age", "", time() - 60);
?>
<html>
<head>
    <title>Stergere cookies folosind PHP</title>
</head>
<body>
    <?php
        if (isset($_COOKIE["name"])) {
            echo "Bine ai venit " . $_COOKIE["name"] . "!<br>";
        } else {
            echo "Imi pare rau, dar nu te cunosc!";
        }
    ?>
</body>
</html>
```

8.3 Sesiunea

O cale alternativă la stocarea informațiilor în cookies este aceea de a stoca datele în sesiuni. O sesiune creează un fișier într-un director temporar pe server unde datele necesare fiecărui client sunt stocate în variabile specifice. Aceste date vor fi disponibile pentru toate paginile site-ului cât timp clientul vizitează site-ul.

Locația fișierului temporar este determinată de setarea din fișierul de configurare **php.ini** de pe server. Înainte de folosirea unei sesiuni trebuie să fim siguri că setam calea.

Atunci când o sesiune începe, PHP prima dată creează un identificator unic pentru acea sesiune reprezentat de obicei de un șir de caractere cu o lungime de 32 valori precum:

3c7foj34c3jj973hjkop2fc937e3443.

Un cookie numit **PHPSESSID** este trimis automat la computerul utilizatorului pentru a stoca șirul unic de identificare a sesiunii și pe client.

Un fișier este creat automat pe server în directorul temporar desemnat și poartă numele identificatorului unic prefixat de **sess_** adică **sess_3c7foj34c3jj973hjkop2fc937e3443**.

Când un script PHP dorește să recupereze valoarea dintr-o variabilă de sesiune, PHP primește automat șirul unic de identificare a sesiunii din cookie-ul PHPSESSID și apoi caută în directorul său temporar fișierul care poartă acel nume și se poate face o validare prin compararea ambelor valori.

Observație: O sesiune se termină atunci când utilizatorul închide browser-ul sau după părăsirea site-ului. De asemenea, de obicei, serverul va încheia sesiunea după o perioadă de timp prestabilită, de exemplu: 30 de minute.

8.3.1 Începerea unei Sesiuni în PHP

O sesiune PHP este pornită cu ușurință prin apelarea funcției **session_start()**. Această funcție verifică mai întâi dacă o sesiune este deja începută și dacă nu este pornită nici una, pornește una. Este recomandat să apelați funcția **session_start()** la începutul paginii.

Variabilele de sesiune sunt stocate într-o matrice asociativă numită **$_SESSION[ ]**. Aceste variabile pot fi accesate pe durata de viață a unei sesiuni.

Următorul exemplu pornește o sesiune, apoi înregistrează o variabilă numită **counter** care este incrementată de fiecare dată când pagina este vizitată în timpul sesiunii.

În exemplul următor reîncărcați pagina de mai multe ori pentru a observa modificarea variabilei **counter** și implicit a mesajului afișat

Observație: Nu uitați să folosiți funcția **isset()** pentru a verifica dacă variabila de sesiune este deja setată sau nu.

```
<?php
   session_start();

   if( isset( $_SESSION['counter'] ) ) {
      $_SESSION['counter'] += 1;
      $msg = "Ati vizitat pagina de " .  $_SESSION['counter'] . "
ori.";
   }else {
      $_SESSION['counter'] = 1;
      $msg = "Ati vizitat pagina o data.";
   }
?>
<html>
   <head>
      <title>Setting up a PHP session</title>
   </head>
   <body>
      <?php  echo ( $msg ); ?>
   </body>
</html>
```

8.3.2 Distrugerea unei Sesiuni PHP

O sesiune PHP poate fi distrusă de funcția **session_destroy()**. Această funcție nu are nevoie de niciun argument și un singur apel poate distruge toate variabilele de sesiune. Dacă doriți să distrugeți o singură variabilă de sesiune, atunci puteți utiliza funcția **unset()** pentru a anula o variabilă transmisă ca parametru. Ștergerea tuturor variabilelor de sesiune se face cu ajutorul funcției **session_unset().**

Mai jos este prezentată modalitatea prin care se poate șterge variabila de sesiune **counter** cu ajutorul funcției **unset()**:

unset($_SESSION['counter**']);**

Un exemplu de ștergere a tuturor variabilelor de sesiune prin apelarea funcției **session_unset()** și a sesiunii este prezentat mai jos:

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
    // șterge toate variabilele de sesiune
    session_unset();

    // șterge sesiunea
    session_destroy();
?>
</body>
</html>
```

8.3.3 Sesiuni fără cookie-uri

Pot exista cazuri în care un utilizator nu permite stocarea cookie-urilor pe computerul său astfel că trebuie apelat la o altă metodă de a trimite ID-ul sesiunii către browser.

Această alternativă constă în utilizarea constantei **SID** care este definită dacă a început sesiunea. Dacă clientul nu a trimis un cookie de sesiune adecvat, acesta are forma **session_name=session_id**. În caz contrar, acesta este un șir gol. Astfel, îl puteți încorpora necondiționat în adrese URL.

Următorul exemplu demonstrează cum să înregistrați o variabilă și cum să faceți legătura corect la o altă pagină folosind SID.

```
<?php
      session_start();

      if (isset($_SESSION['counter'])) {
          $_SESSION['counter'] = 1;
      } else {
          $_SESSION['counter']++;
      }
      $msg = "Ați vizitat această pagină " .  $_SESSION['counter'];
      $msg .= "în această sesiune.";

      echo ($msg);
?>

<p>
    Pentru a continua apăsați pe următorul link <br />

    <a href="nextpage.php?<?php echo htmlspecialchars(SID); ?>">
</p>
```

Teme

1. Realizați o pagină web ce stochează cookies (informații la alegerea voastră) pe calculatorul clientului. Datele stocate vor avea o durată de expirare de 5 min. Verificați dacă informațiile stocate mai sunt disponibile după expirarea celor 5 min.
2. Realizați o pagină ce stochează informații de tip sesiune. Se va stoca numărul de accesări al unei pagini. Accesați aceeași pagină folosind browsere diferite.