Tehnologii Internet

CURSUL 10 – SERVERUL WEB (2)

Universitatea Tehnică ”Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare
Specializarea Tehnologia informației

Cuprins

1. Limbajul PHP (continuare)
2. Protocolul HTTP (continuare)

1. Limbajul PHP

1.1. Formulare PHP (continuare)

1.2. Fișiere în PHP

1.3. Tratarea erorilor în PHP

1.1. Formulare PHP

- Exemplu de cod/pagina PHP

La server	La client
<!DOCTYPE html> <html> <body> Tehnologii internet <?php echo ' Hello World! '; ?> </body> </html>	<!DOCTYPE html> <html> <body> Tehnologii internet Hello World! </body> </html>

1.1. Formulare PHP

- $_GET vs. $_POST
- Validarea datelor
- Afișarea unor mesaje de eroare dacă datele nu sunt valide
- Folosirea aceleiași pagini PHP pentru afișarea unui formular și pentru procesarea acestuia

1.1. Formulare PHP

GET vs. POST

GET /form.php?**nume=Ion&prenume=Pop** HTTP/1.1
Host: aatuiasi.appspot.com

POST /form.php HTTP/1.1
Host: aatuiasi.appspot.com
nume=Ion&prenume=Pop

1.1. Formulare PHP

GET vs. POST

Cererile GET

- Folosite pentru a lua date de la server
- Pot fi adăugate în cache și ca ca bookmark-uri
- Rămân în istoricul browser-elor
- Sunt mai puțin sigure
- Au o limitare dată de numărul maxim de caractere dintr-un URL (aprox. 2048)
- Sunt mai eficiente dpdv. AJAX: prin POST se trimit două cereri (una cu header-e și alta cu datele)

1.1. Formulare PHP

GET vs. POST

Cererile POST

- Folosite pentru a modifica date la server
- Folosite când se dorește trimiterea unui volum mare de date (URL-ul este limitat la aprox. 2048) sau când se trimit date private (e.g., parole)
- Pot fi trimise date binare (la GET sunt permise doar caractere ASCII)
- Sunt mai sigure deoarece parametrii nu sunt stocați în istoricul browser-elor sau în cache

1.1. Formulare PHP

- Folosirea aceleiași pagini PHP pentru afișarea unui formular și pentru procesarea acestuia

```
<form method="post" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

1.1. Formulare PHP

Validarea datelor

- Afișarea unor mesaje de eroare dacă datele nu sunt valide
- Validare la client și/sau validare la server
- Exemple de validări:
  - Numele și prenumele conțin doar litere, spații și –
  - Adresă de email validă (@ .)
  - Sex: masculin / feminin
- Câmpuri obligatorii
- Dacă avem butoane radio, trebuie validare? Unde?

1.1. Formulare PHP

Securitatea - Cross Site Scripting (XSS)

- XSS – vulnerabilitate de securitate în aplicațiile web
- Permite atacatorilor să injecteze scripturi client-side în paginile web vizualizate de alți utilizatori
- http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
-
```
<form method="post"
action="test_form.php/">
<script>alert('hacked')</script>
```

1.1. Formulare PHP

Validarea datelor folosind PHP

- Convertirea caracterelor speciale (&, ‘, “, <, >)

- `<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">`
- `<form method="post" action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">`

1.1. Formulare PHP

Validarea datelor folosind PHP

- Definirea unei funcții care să scoată elementele ce pot duce la vulnerabilități din punctul de vedere al securității:

```
function test_input($data) {
   $data = trim($data);
   $data = stripslashes($data);
   $data = htmlspecialchars($data);
   return $data;
}
```

1.1. Formulare PHP

Validarea datelor folosind PHP

- Validarea la server

```
if (empty($_POST["nume"])) {
    $eroare = "Numele trebuie completat";
} else {
    $nume = test_input($_POST["nume"]);
}
```

- Afișarea mesajelor de eroare

```
<span class="eroare"><?php echo $eroare;
?></span>
```

1.1. Formulare PHP

Validarea datelor folosind PHP

- Validarea la server
  - `preg_match(`*`pattern, șir`*`)` – expresii regulate
  - `filter_var(`*`șir, filtru, opțiuni`*`)` – validează șirul de caractere în funcție de filtru și de opțiuni
- Exemple de filtre folosite de funcția `filter_var`
  - FILTER_VALIDATE_EMAIL
  - FILTER_SANITIZE_EMAIL
  - FILTER_VALIDATE_INT – min, max
  - FILTER_VALIDATE_REGEXP – regexp
  - FILTER_VALIDATE_URL

1.1. Formulare PHP

Interpretarea ca vector în PHP a datelor primite de la un formular HTML

```
<input name="unVector[]" />
<input name="unVector[]" />
<input name="unVector[nume]" />
<input name="unVector[email]" />
```

- Variabila `$_GET['unVector']` va fi un vector cu cheile: 0, 1, 'nume' și 'email'
- Dacă tag-ul select permite selecția multiplă, atunci neapărat trebuie specificat numele astfel:

```
<select name="optiune[]" multiple="yes">
```

1.3. Tratarea erorilor în PHP

Terminarea scriptului

- `die(mesaj);`
- `exit(mesaj);`
- Este afișat un mesaj și apoi scriptul este terminat
- Exemplu
  - `$f = fopen("f.txt", "w") or die("eroare la deschidere");`

1.3. Tratarea erorilor în PHP

Crearea unei funcții care să trateze eroarea

- *functieDeEroare*(*nivel, mesaj, fișier, linie, context*)
- Primii doi parametri sunt obligatorii
- Nivel:

Valoare	Constantă
2	E_WARNING
8	E_NOTICE
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
4096	E_RECOVERABLE_ERROR
8191	E_ALL

1.3. Tratarea erorilor în PHP

Specificarea funcției care să trateze eroarea

- `set_error_handler("functieDeEroare", nivel);`
- Parametrul nivel este opțional

```
<?php
function functieDeEroare($errno, $errstr) {
  echo "<b>Eroare:</b> [$errno] $errstr";
}
set_error_handler("functieDeEroare");
echo($test);
?>
//Eroare: [8] Undefined variable: test
```

1.3. Tratarea erorilor în PHP

Declanșarea unei erori

- `trigger_error(`*`mesaj, nivel`*`);`
- Parametrul nivel este opțional

```
<?php
set_error_handler("functieDeEroare");
$test=3;
if ($test>1)
     trigger_error("test > 1",
          E_USER_WARNING);
?>
//Eroare: [512] test > 1
```

1.3. Tratarea erorilor în PHP

Logarea unei erori

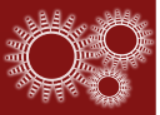
- `error_log(`*`mesaj, tip_mesaj, destinație, extra_headere`*`);`
- Permite logarea unei erori folosind sistemul de logare PHP sau trimiterea pe mail a erorii respective

1.3. Tratarea erorilor în PHP

Excepții

- Instrucțiunile `try, throw, catch`
- Exact ca în limbajul Java

- Mai multe detalii la cursul referitor la PHP-OO



2. Protocolul HTTP

2.1. Cererea și răspunsul HTTP

2.2. Proxy

2.3. Gateway

2.4. Tunel

2.5. Cache

2.6. Cookie

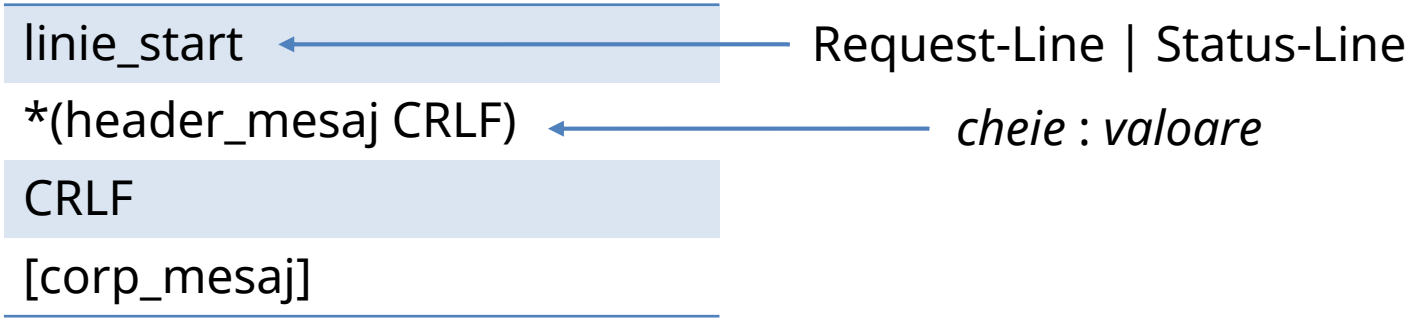
2.7. Sesiune

2.8. Cookie vs. Sesiune

2.1. Cererea și răspunsul HTTP

Cerere – Răspuns (Request – Response)

- Un mesaj HTTP (cerere sau răspuns) trebuie să aibă următoarea formă:



2.2. Proxy

- Aplicație intermediară care este în același timp server și client cu rolul a răspunde cererilor altor clienți
- În Request-Line valoarea pentru request-URI este adresa completă

2.2. Proxy

Header-e HTTP specifice cererii trimise către un proxy

- *Proxy-Authorization*
  - e.g., Proxy-Authorization: Basic dXNlcjpwYXJvbGE=
- *X-Forwarded-For, X-Forwarded-Host, X-Forwarded-Proto* - adresa, domeniul, respectiv, protocolul clientului inițial care a făcut cererea
- *Max-Forwards* - numărul maxim de noduri (proxy sau gateway) prin care poate să treacă cererea
- *Via* – proxy-urile prin care a trecut cererea

2.2. Proxy

Header-e HTTP specifice răspunsului primit de la un proxy

- *Proxy-Authenticate*
  - e.g., Proxy-Authenticate: Basic
- *Via* – proxy-urile prin care a trecut răspunsul

2.2. Proxy

- Dacă un proxy necesită autentificare, atunci răspunsul este forma:
  - 407 Proxy Authentication Required
- Header-ele *Proxy-Authorization* (cerere) și *Proxy-Authenticate* (răspuns) sunt header-e single-hop (nu sunt transmise mai departe)
- Header-ele *Authorization* (cerere) și *WWW-Authenticate* (răspuns) sunt header-e end-to-end (sunt transmise nemodificate prin serverele intermediare)

2.2. Proxy

Tipuri de proxy

- *Transparent* – cererea și răspunsul nu sunt modificate (cu excepția elementelor necesare autentificării și identificării serverului proxy)
- *Non-transparent* – cererea și/sau răspunsul sunt modificate

2.2. Proxy

- Header-e HTTP care pot fi modificate de proxy

User-Agent	Accept-Encoding	Via
Accept	Accept-Language	
Accept-Charset	X-Forwarded-For	

- Dacă o cerere conține header-ul

```
Cache-Control: no-transform
```

serverul proxy nu trebuie să modifice cererea (proxy transparent)

2.2. Proxy

Utilitatea serverelor proxy

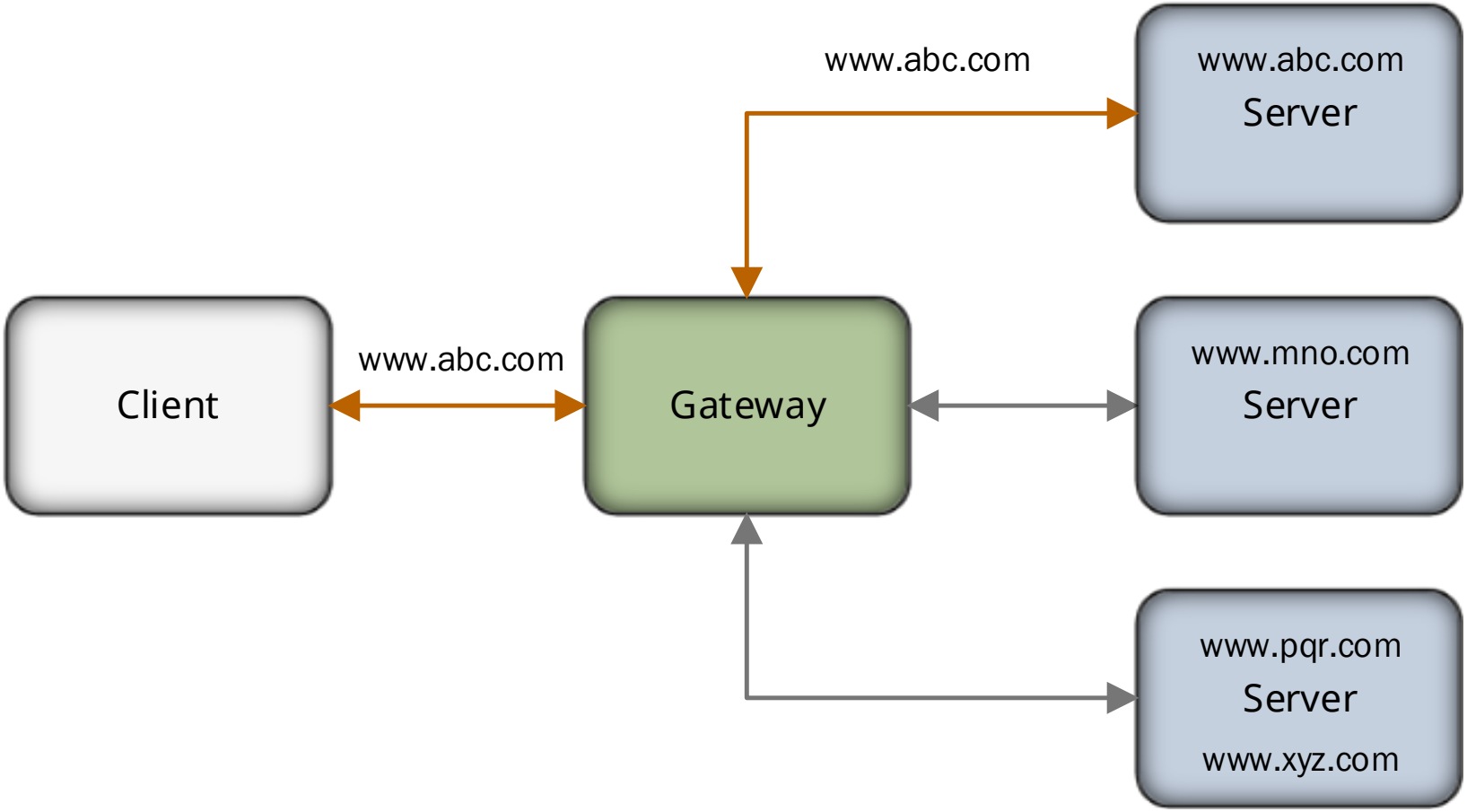
- Securitate (scanarea conținutului împotriva malware-ului)
- Anonimitate
- Traducerea răspunsului
- Logare
- Autentificare
- Caching
- Filtrarea conținutului (e.g, control parental)
- Accesarea unor adrese care, în mod normal, sunt blocate sau filtrate

2.3. Gateway

Gateway

- Server care este un intermediar pentru alte servere
- Clientul care face cererea, de obicei, nu știe că trimite cererea la un gateway sau la un server web
- Spre deosebire de un proxy, destinatarul cererii este gateway-ul

2.3. Gateway



2.4. Tunel

Tunel

- Program intermediar care face legătura dintre două conexiuni
- Datele sunt transferate între cele două conexiuni fără a fi modificate sau monitorizate de tunel
- Tunelul există doar atât timp cât ambele conexiuni sunt active

2.5. Cache

Cache

Mecanism de stocare temporară a resurselor web

- Avantaje:
  - Reducerea traficului pe rețea
  - Reducerea încărcării serverului
  - Reducerea timpului de răspuns la cererea clientului
- Metode:
  - Eliminarea trimiterii unor cereri la server
  - Trimiterea de către server a unor răspunsuri ”mai scurte”

2.5. Cache

- O entitate din cache este considerată validă dacă entitatea nu a fost modificată după ce a fost stocată în cache
- Header-e utilizate de mecanismul de cache
  - Cache-control (la client și la server)
  - (1) Last-Modified (la client și la server)
  - (1) ETag (la server)
  - If-Match, If-None-Match, If-Range (la client)
  - If-Modified Since (la client)
  - If-Not-Modified-Since (la client)
  - Warning

2.5. Cache

Modalități de control al unui cache

1. Restricții cu privire la ce răspunsuri pot fi considerate pentru a fi stocate în cache
2. Restricții cu privire la ce poate fi efectiv stocat într-un cache
3. Specificarea unui mecanism de expirare a cache-ului
4. Controlarea revalidării și reîncărcării cache-ului
5. Controlarea posibilității de a modifica răspunsul înainte de a fi stocat în cache
6. Extinderea sistemului cache

2.5. Cache

Header-ul `Cache-Control`

Cerere (client - user-agent)	Răspuns (server)
(4) no-cache	(1) public
(2) no-store	(1) private
(3) max-age = *secunde*	(1) no-cache
(3) max-stale [= *secunde*]	(2) no-store
(3) min-fresh = *secunde*	(5) no-transform
(5) no-transform	(4) must-revalidate
(4) only-if-cached	(4) proxy-revalidate
(6) cache-extension	(3) max-age = *secunde*
	(3) s-max-age = *secunde*
	(6) cache-extension

2.6. Cookie

Cookie

- Informație trimisă de serverul web către client (browser) care este stocată de client și trimisă la server când utilizatorul accesează site-ul respectiv
- Fiecare cookie are un timp de expirare dat în secunde (e.g., time()+86400)
- Dacă timpul de expirare este omis sau este 0, cookie-ul va expira la sfârșitul sesiunii (închiderea browser-ului)

2.6. Cookie

Utilizare

- Managementul sesiunilor
- Personalizarea site-ului
- Tracking – urmărirea comportamentului utilizatorului pe un anumit site

2.6. Cookie

Creare și utilizarea

- Răspunsul primit de la serverul web conține header-ul: `Set-Cookie`
- Cererile următoare către serverul web respectiv vor conține header-ul: `Cookie`

Exemplu:

- Server –> Client
  - Set-Cookie: SID=31d4d96e407aad43
- Client –> Server
  - Cookie: SID=31d4d96e407aad43

2.6. Cookie

Sintaxa Set-Cookie

```
Set-Cookie: nume=valoare [; Expires=dată]
[; Max-Age=nrSecunde] [; Domain=domeniu]
[; Path=cale] [; Secure] [; HttpOnly]
```

Exemplu

```
Set-Cookie: CP3=1; expires=Sat, 10-Jan-
2015 02:38:04 GMT; path=/;
domain=.scorecardresearch.com
```

2.6. Cookie

Sintaxa Cookie

```
Cookie: nume=valoare [; nume2=val2 […]]
```

Exemplu

```
Cookie: UID=6fb0703e-81.196.26.146-
1367091656; UIDR=1398630478
```

2.6. Cookie

Header-ele DNT și TSV

- Interzicerea colectării datelor referitoare la un utilizator
- Draft W3C

DNT (Do Not Track) - header specific cererii

- Poate avea două valori: 1 și 0

TSV (Tracking Status Value) - header specific răspunsului

- Valoarea este o literă

2.6. Cookie

Alternative la utilizarea cookie-urilor

- Transmiterea informațiilor prin URL (query_string)
- Tracking prin adresa IP a utilizatorului
- Formulare cu câmpuri ascunse
- Autentificare HTTP
- Header-ul Etag
- Web storage
- Cache-ul browser-ului

2.6. Cookie

Proprietatea cookie din DOM (JavaScript)

- `document.cookie`
- Permite setarea unor cookie-uri prin specificarea unui șir de caractere cu sintaxa de la header-ul Set-Cookie
- Returnează un șir de caractere care reprezintă cookie-urile din documentul curent având sintaxa de la header-ul Cookie
- Pentru a șterge un cookie trebuie setată o dată de expirare din trecut

2.6. Cookie

Cookie-uri cu PHP

- Crearea, modificarea și ștergerea unui cookie se fac cu ajutorul funcției `setcookie`

```
setcookie(nume, valoare, datăExpirare,
cale, domeniu, secure, httponly);
```

- Cookie-urile se pot obține cu ajutorul variabilei globale `$_COOKIE[nume]`

2.7. Sesiune

Sesiune

- Conversație între un client și server
- Secvență de cereri și răspunsuri
- HTTP este un protocol "fără stare" (en., stateless)

Identificator de sesiune

- en., *session ID* sau *session token*
- Șir de caractere, generat aleatoriu sau de o funcție hash, folosit pentru a identifica o sesiune
- Este trimis prin cookie-uri și/sau formulare HTML

2.7. Sesiune

Sesiuni PHP

- Când sesiunea este creată, id-ul de sesiune este trimis clientului respectiv (`session_start()`)
- Id-ul este stocat la client într-un cookie cu numele `PHPSESSID`
- La fiecare cerere cookie-ul respectiv este trimis la server
- Dacă că cookie-urile nu sunt activate se folosește atributul `PHPSESSID` în URL
- O sesiune se termină și datele sunt stocate atunci când PHP termină de executat script-ul sau când este apelată funcția `session_write_close()`

2.7. Sesiune

Sesiuni PHP

- Datele asociate unei sesiuni se reține la server

- `session_id(`*`id`*`);` – setează sau returnează id-ul de sesiune
- `session_name(`*`nume`*`);` – setează sau returnează numele sesiunii (implicit: `PHPSESSID`)
- `SID` – constantă definită la pornirea sesiunii, de forma `"nume=id"`

2.7. Sesiune

Sesiuni PHP

- `$_SESSION` – variabilă globală folosită pentru a manipula variabilele de sesiune
- `session_unset();` – șterge toate variabilele de sesiune
- `session_destroy();` – distruge sesiunea

2.7. Sesiune

```php
<?php
    // context privat
    session_name('Privat');
    session_start();
    $private_id = session_id();
    $varPrivata = $_SESSION['varP'];
    session_write_close();
    // context global
    session_name('Global');
    session_id('TEST');
    session_start();
    $varGlobala = $_SESSION['varG'];
    session_write_close();
?>
```

Adaptare după sursa: http://php.net/manual/en/session.examples.php

2.7. Sesiune

```
<html>
<body>
   <h1>Test Global: <?=++$varGlobala?></h1>
   <h1>Test Privat: <?=++$varPrivata?></h1>
   <h1>ID Privat: <?=$idPrivat?></h1>
   <h1>ID Global: <?=session_id()?></h1>
   <pre><?php print_r($_SESSION);?></pre>
</body>
</html>
```

Adaptare după sursa: http://php.net/manual/en/session.examples.php

2.7. Sesiune

```php
<?php
    // salvarea valorilor - context privat
    session_name('Privat');
    session_id($idPrivat);
    session_start();
    $_SESSION['varP'] = $varPrivata;
    session_write_close();
    // salvarea valorilor - context global
    session_name('Global');
    session_id('TEST');
    session_start();
    $_SESSION['varG']=$varGlobala;
    session_write_close();
?>
```

Adaptare după sursa: http://php.net/manual/en/session.examples.php

2.8. Cookie vs. Sesiune

Variabilele din cookie vs. variabilele de sesiune

	Var. cookie	Var. sesiune
Stocarea datelor	La client	La server, într-un director public temporar*
Durata de viață	Specificată. Orice durată (secunde, ore, zile, ani, ...)	Predeterminată* (e.g., închiderea browser-ului)
Limitări	Cel puțin 300 cookie-uri, 4kb/cookie, 20 cookie-uri/domeniu**	Nelimitat*
Siguranța datelor	Nesigure. Informațiile de la client pot fi alterate	Sigure. Clientul nu are acces direct la ele
Comunicarea cu alte web servere	Nici o problemă	Problemă, dacă nu este comunicare între web servere sau shared storage

* - depinde de configurarea serverului PHP
** - conform RFC 2109

Bibliografie

- http://www.w3schools.com/php/default.asp
- http://php.net/manual/en/
- http://www.w3schools.com/tags/ref_httpmethods.asp
- http://www.diffen.com/difference/GET_%28HTTP%29_vs_POST_%28HTTP%29
- http://computer.howstuffworks.com/cookie.htm
- http://www.nczonline.net/blog/2009/05/05/http-cookies-explained/
- http://www.w3schools.com/js/js_cookies.asp
- http://php.net/manual/en/book.session.php
- http://php.net/manual/en/session.idpassing.php
- http://php.net/manual/en/ref.session.php
- http://php.net/manual/en/session.configuration.php
- http://tools.ietf.org/html/rfc2616
- http://tools.ietf.org/html/rfc2617
- http://tools.ietf.org/html/rfc6265