

## Plan

## Programmation Internet

### Cours 8

kn@lri.fr  
<http://www.lri.fr/~kn>

- 1 Systèmes d'exploitation (1/2) ✓
- 2 Systèmes d'exploitation (2/2) ✓
- 3 Réseaux, TCP/IP ✓
- 4 Web et HTML ✓
- 5 CSS ✓
- 6 PHP : Introduction ✓
- 7 PHP : Sessions et persistance
  - 7.1 En-tête de requêtes HTTP
  - 7.2 Cookies
  - 7.3 Sessions

## Retour sur le protocole HTTP

### ■ Client :

```
GET /~kn/index.html HTTP/1.1
Host: www.lri.fr
```

### ■ Serveur :

```
HTTP/1.1 200 OK
Server: nginx/1.4.1 (Ubuntu)
Date: Sun, 17 Nov 2013 16:44:48 GMT
Content-Type: text/html
Content-Length: 2038
```

```
<html>
  <head><title>Homepage</title> </head>
  <body>
    ...
```

```
} ← code de retour
} ← type de contenu
} ← longueur du contenu
} ← contenu (2038 octets)
```

## Retour sur le protocole HTTP (2)

### ■ Client :

```
GET /~kn/fichier.pdf HTTP/1.1
Host: www.lri.fr
```

### ■ Serveur :

```
HTTP/1.1 200 OK
Server: nginx/1.4.1 (Ubuntu)
Date: Sun, 17 Nov 2013 16:44:48 GMT
Content-Type: application/pdf
Content-Length: 103449
```

```
%PDF-1.2
%
8 0 obj
<</Length 9 0 R/Filter /FlateDecode>>
stream
.....
```

## Modifier le content-type en PHP

Fichier notes\_csv.php:

```
<?php
header('Content-type: application/csv');
header('Content-Disposition: attachment; filename="notes.csv"');
echo "Nom, Note\n";
foreach ($NOTES as $nom => $note)
    echo $nom . ", " . $note . "\n";
?>
```

### ⚠ Attention!

- Les appels à la fonction `header()` doivent se trouver **avant** le premier `echo()` du code PHP
- Le code PHP doit générer (avec `echo()`) du contenu compatible avec le type annoncé (et pas du HTML)

## Quelques en-tête utiles

En tête utilisés par le serveur dans ses réponses

**Content-type** : type MIME du contenu envoyé par le serveur  
**Content-Disposition** : permet de mentionner un nom de fichier : `attachment; filename="foobar.baz"`  
**Cache-Control** : permet de forcer le client à retélécharger la page: `no-cache, must-revalidate`  
**Last-Modified** : date de dernière modification du contenu demandé

En tête utilisés par le client dans ses requêtes

**Range** : permet de ne récupérer qu'un intervalle d'octets donné dans un fichier:  
`bytes=500-999`

...

## Retour sur le protocole HTTP (3)

On rappelle que HTTP est un protocole *stateless* (sans état, i.e. le serveur Web ne conserve pas d'information entre les connexions). Quel problème cela pose-t-il ?

- Pas de partage d'information entre plusieurs pages
  - Pas de mécanisme de reprise sur panne
  - Pas de persistance de l'information
  - Pas d'authentification (impossible de savoir que deux connexions successives ont été faites par le même client)
- ⇒ difficile de réaliser une « application » moderne répartie sur plusieurs pages

## Plan

- 1 Systèmes d'exploitation (1/2) ✓
- 2 Systèmes d'exploitation (2/2) ✓
- 3 Réseaux, TCP/IP ✓
- 4 Web et HTML ✓
- 5 CSS ✓
- 6 PHP : Introduction ✓
- 7 PHP : Sessions et persistance
  - 7.1 En-tête de requêtes HTTP ✓
  - 7.2 Cookies
  - 7.3 Sessions

## Cookies

Un *cookie* est un packet de données envoyé par le serveur, stocké par le client (navigateur Web) et renvoyé au serveur lors d'une nouvelle connexion. Les propriétés d'un cookie sont:

**Son nom :** une chaîne de caractères

**Sa valeur :** une chaîne de caractères

**Sa durée de vie :** jusqu'à la fin de la « session » ou pour une période donnée

**Son domaine :** Le nom du site web émetteur du cookie

**Son chemin :** Le sous-répertoire (par rapport à la racine du site) pour lequel le cookie est valide

⚠ **Attention!** seul le domaine qui a déposé le cookie est capable de le relire

## Cookies en PHP

Créer ou mettre à jour un cookie sur le client:

```
setcookie($nom, $val, $date);
```

**\$nom :** nom du cookie

**\$val :** valeur du cookie

**\$date :** date d'expiration en secondes depuis *epoch* (1<sup>er</sup> janvier 1970 00:00:00) ou NULL pour une expiration automatique.

(on peut récupérer le nombre de secondes depuis *epoch* avec la fonction `time()`).

Exemple:

```
setcookie("mon_cookie", "42", time() + 3600 * 24 * 30);
```

## Cookies en PHP

On peut récupérer la valeur d'un cookie depuis PHP:

```
$_COOKIE["mon_cookie"]
```

Un cookie "foo" existe (*i.e.* a été défini auparavant) si une entrée correspondante existe dans le tableau global `$_COOKIE`. On peut tester qu'une entrée existe dans un tableau avec `isset()`.

⚠ **Attention!**

- On ne peut pas écrire dans `$_COOKIE` (par exemple `$_COOKIE["foo"] = 42`), il faut utiliser `setcookie()`.
- `setcookie()` utilise `header()` et doit donc être appelé avant le premier `echo()` du fichier.
- Pour effacer un cookie, on peut lui donner une date d'expiration antérieure à l'instant présent (0 par exemple)

## Avantages et inconvénients des cookies

- + stockage persistant
- + interface simple d'utilisation (une variable pour la lecture et `setcookie` pour l'écriture)
- limité en taille
- limité en nombre par domaine
- type de donnée limité à des chaînes (on ne peut pas stocker un tableau PHP par exemple)
- +/- stocké sur le client

## Plan

- 1 Systèmes d'exploitation (1/2) ✓
- 2 Systèmes d'exploitation (2/2) ✓
- 3 Réseaux, TCP/IP ✓
- 4 Web et HTML ✓
- 5 CSS ✓
- 6 PHP : Introduction ✓
- 7 PHP : Sessions et persistance
  - 7.1 En-tête de requêtes HTTP ✓
  - 7.2 Cookies ✓
  - 7.3 Sessions

## Sessions

Une **session HTTP** est un ensemble de requêtes/réponses HTTP entre un serveur et un **même** client.

Exemple d'un sondage en ligne:

1. Le visiteur arrive sur la page `q1.php` en cliquant sur le lien « commencer le sondage » (**Début de session**)
2. Sur `q1.php`, l'utilisateur coche des choix dans un formulaire et appuie sur un bouton de soumission qui l'envoie sur `q2.php`
3. ...
4. Sur `q10.php`, l'utilisateur coche des choix dans un formulaire et appuie sur un bouton de soumission qui l'envoie sur `resultat.php`
5. Sur `resultat.php`, le résultat global du sondage (% par question, nombre de participants jusqu'à présent etc...) est affiché (**Fin de session**)

## Variables de session

Pour programmer une application Web, on souhaite avoir accès à des **variables de session** c'est à dire des variables qui sont:

- Globale au serveur, et accessibles depuis plusieurs pages PHP différentes
- Spécifiques à un « utilisateur » (c'est à dire à une session particulière)

Les variables de sessions sont donc propres à chaque client et persistent le temps de la session (le temps de session est décidé par le serveur)

## Variables de session en PHP

On initie une session avec la fonction:

```
session_start();
```

Une fois appelée, la variable `$_SESSION` contient un tableau que l'on peut utiliser entre plusieurs pages. Les valeurs contenues dans le tableau persistent jusqu'à la fin de la session. Une session se termine:

- Quand le client se déconnecte
- Après un certain temps (« votre session a expiré, veuillez vous reconnecter »)
- Quand le code PHP appelle `session_end()`;

⚠ **Attention!** `session_start()` doit être appelé avant le premier echo du fichier.

## Variables de session en PHP (2)

## Avantages et inconvénients des session

```
<?php /* Fichier page1.php */
  session_start();
  $_SESSION["Valeur"] = 42;
?>
<html>
  <body>
    Veuillez cliquer sur le <a href="page2.php">lien</a>
  </body>
</html>

<html>
  <body>
    La valeur est <?php echo $_SESSION["Valeur"]; ?>
    <!-- affiche 42 -->
  </body>
</html>
```

- + Informations stockées sur le serveur
- + Pas de limite de taille
- + Pas limité à des chaînes de caractères
- Valeurs perdues en fin de session
- **Nécessite des cookies**

## Sessions PHP: détails d'implantation

### Coté client

Connexion à une page PHP (envoi du cookie ("php\_ssid", "12345"))

### Coté serveur (PHP)

```
session_start();
- génération d'un ID unique "12345"
- dépôt d'un cookie "php_ssid", valeur
"12345", durée 10 minutes - création
dans un tableau global d'une entrée:
$_GLOBAL["12345"] = Array();

$_SESSION =
$_GLOBAL[$_COOKIE["php_ssid"]]
```

## Dans la vraie Vie™

Mélange de variables de sessions, cookies et bases de données.  
Scénario réaliste: site de commerce en ligne

- Login/mot de passe stocké dans une **BD**
- Panier courant stocké dans une **variable de session**
- Login, date de dernière visite, dernière page visitée stockés dans un **en cookie**