Programmation Internet

Cours 7

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

Plan

1 Réseaux, TCP/IP ✔
2 Web et HTML ✔
3 CSS ✔
4 PHP : Introduction ✔
5 PHP : expressions régulières, fichiers, sessions
- 5.1 Manipulation des chaînes et expressions régulières
- 5.2 Manipulation de fichiers
- 5.3 En-tête de requêtes HTTP
- 5.4 Cookies
- 5.5 Sessions

Quelques fonctions utilitaires sur les chaînes

`explode($delim, $entree)` : **Découpe la chaîne** `$entre e` **suivant la sous-chaîne** `$delim` **et renvoie les morceaux dans un tableau.**

`implode($delim, $tab)` : **Réunit les chaînes se trouvant dans le tableau** `$tab` **en les séparant par la chaîne** `$delim`.

`ltrim($entree)` : **Retire les caractères blancs en début de chaîne.**

`rtrim($entree)` : **Retire les caractères blancs en fin de chaîne.**

`trim($entree)` : **Retire les caractères blancs en début et en fin de chaîne.**

`htmlspecialchars($entree)` : **convertit les caractères** &, ", ', < **et** > **en** `&`, `&quot;`, `&apos;`, `<` **et** `>`.

Expressions régulières : syntaxe

Les expressions régulières de PHP sont au format PCRE (Perl Common Regular Expressions) `'/r/'` **où** `r` **est une expression de la forme:**

r	::=	a	(un caractère)
	|	.	(n'importe quel caractère)
	|	r_1 | r_2	(r_1 ou r_2)
	|	r?	(r répétée au plus 1 fois)
	|	r*	(r répétée 0 fois ou plus)
	|	r+	(r répétée 1 fois ou plus)
	|	[c_1 … c_n]	(un caractère parmis c_1, …, c_n)
	|	[c_1-c_n]	(un caractère parmis c_1, …, c_n)
	|	[^c_1 … c_n]	(un caractère sauf c_1, …, c_n)
	|	[^c_1-c_n]	(un caractère sauf c_1, …, c_n)
	|	^	(début de texte)
	|	$	(fin de texte)
	|	(r)	(r elle même)

Expressions régulières : recherche

`preg_match($regexp, $chaine)`

renvoie 1 si une sous-chaine de `$chaine` correspond à `$regexp`, 0 si aucune sous-chaine ne correspond et `FALSE` en cas de problème (attention, utiliser `===` pour tester le résultat).

```php
<?php
    $chaine = "ABCDEFABCDEF";
    echo preg_match('/ABC/', $chaine);       // affiche 1
    echo preg_match('/DEF/', $chaine);       // affiche 1
    echo preg_match('/^ABC/', $chaine);      // affiche 1
    echo preg_match('/^DEF/', $chaine);      // affiche 0
    echo preg_match('/ABC$/', $chaine);      // affiche 0
    echo preg_match('/DEF$/', $chaine);      // affiche 1
    echo preg_match('/(ABC...)+/', $chaine); // affiche 1
    echo preg_match('/[^A-Z]+/', $chaine);   // affiche 0
    echo preg_match('/[^A-Z]*/', $chaine);   // affiche 1 !
    echo preg_match('/^[^A-Z]*$/', $chaine); // affiche 0
?>
```


Expressions régulières : substitution

`preg_replace($regexp, $motif, $chaine)`

recherche toutes les sous-chaînes de `$chaine` reconnues par `$regexp` et les remplace par `$motif`. Ce dernier peut contenir `$i` pour référencer le $i^{ème}$ groupe de parenthèses

```php
<?php
    $chaine = "10-31-1981";
    $reg1 = "/([0-9]+)-([0-9]+)-([0-9]+)/";
    echo preg_replace($reg1, "$2/$1/$3", $chaine);
    // affiche 31/10/1981

    $reg2 = "/1/";
    echo preg_replace($reg2, "toto", $chaine);
    // affiche toto0-3toto-toto98toto

    $reg3 = "/[0-9]([0-9]*)/"
    echo preg_replace($reg3, "$1", $chaine);
    // affiche 0-1-981 (* déplie la regexp le plus possible)
?>
```


Expressions régulières : séparation

`preg_split($regexp, $chaine)`

renvoie un tableau des sous-chaine de `$chaine` séparées par `$motif` (équivalent à `explode` pour des `$regexp` constantes).

```php
<?php
        $chaine = "Une phrase, c'est plusieurs mots.";
        print_r (preg_split("/[ ,.']+/", $chaine));
        //Affiche:
        //Array ( [0] => Une [1] => phrase [2] => c [3] => est
        //        [4] => plusieurs [5] => mots [6] => )
?>
```


Expressions régulières : recherche exhaustive

`preg_match_all($regexp, $chaine, &$resultat)`

`&$resultat` est un tableau passé par référence. Après l'appel, `$resultat[0]` contient un tableau avec toutes les sous-chaines reconnues et `$resultat[i]` contient tous les résultats reconnus par le $i^{ème}$ groupe de parenthèses. Renvoie le nombre de chaines trouvées (*i.e.* la longueur de `$resultat[0]`).

```php
<?php
    $res = array();
    $chaine = "ABC ACD AEF AB DEF";
    echo preg_match_all("/A([A-Z]*)/", $chaine, $res);
    //Affiche 4
    print_r ($res);
    /*Affiche
    Array ( [0] =>
              Array ( [0] => ABC [1] => ACD [2] => AEF [4] => AB)
            [1] =>
              Array ( [0] => BC  [1] => CD  [2] => EF  [4] => B)
            )
*/?>
```

Plan

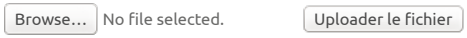
1 Réseaux, TCP/IP ✔
2 Web et HTML ✔
3 CSS ✔
4 PHP : Introduction ✔
5 PHP : expressions régulières, fichiers, sessions
  5.1 Manipulation des chaînes et expressions régulières ✔
  5.2 Manipulation de fichiers
  5.3 En-tête de requêtes HTTP
  5.4 Cookies
  5.5 Sessions

Envoi d'un fichier au serveur (1/2)

On utilise la méthode `post` pour les formulaires. Les valeurs sont envoyées dans la requête HTTP (et non pas encodées dans l'URL)

```
<form action="cible.php" method="post"
      enctype="multipart/form-data">
      <input type="file" name="fichier" size="20"/>
      <button type="sumbit">Uploader le fichier</button>
</form>
```

Apperçu:



Coté serveur, la variable `$_FILES` est définie. `$_FILES["fichier"]` contient un tableau avec des informations sur le fichier envoyé. Les autres champs (par exemple valeur d'un champ texte) sont stockés dans la variable `$_POST` (au lieu de `$_GET`).

Envoi d'un fichier au serveur (2/2)

Étant donné un formulaire avec un champ *input* de type *file* et de nom `"fichier"` on a accès aux information suivantes:

`$_FILES["fichier"]["error"]` : Code d'erreur (0 si tout c'est bien passé, > 0 si une erreur s'est produite. Les autres champs ne sont définis que si "error" vaut 0).

`$_FILES["fichier"]["tmp_name"]` : Nom du fichier temporaire sur le serveur où a été sauvegardé le contenu du fichier envoyé

`$_FILES["fichier"]["name"]` : Nom original du fichier

`$_FILES["fichier"]["size"]` : Taille du fichier

`$_FILES["fichier"]["type"]` : Le type MIME du fichier

Ouverture d'un fichier

```
fopen($nomfichier, $mode)
```

Ouvre un fichier dont on donne le nom avec un *mode* particulier. Les modes sont: `"r"` (lecture), `"r+"` (lecture/écriture), `"w"` (écriture), `"w+"` (lecture/écriture, création si non-existant), `"a"` (écriture, ajout à la fin si existant), `"a+"` (lecture/écriture, création si non-existant, ajout à la fin si existant). `fopen` renvoie un descripteur de fichier que l'on peut utiliser pour manipuler le fichier.

Lecture/écriture/fermeture d'un fichier

```
fread($desc, $taille)
```

Lit au maximum `$taille` **caractères dans un fichier dont le descripteur (renvoyé par** `fopen`**) est** `$desc`**. La fonction renvoie un chaine d'au plus** `$taille` **caractères ou** `FALSE` **en cas d'erreur.**

```
fwrite($desc, $chaine)
```

Écrit la chaine de caractères à la position courante dans le fichier dont le descripteur est `$desc`**. Renvoi le nombre d'octets écrits ou** `FALSE` **en cas d'erreur.**

```
fclose($desc)
```

Ferme le fichier dont le descripteur est `$desc`

Déplacement dans un fichier

```
fseek($desc, $offset, $orig)
```

Déplace le pointeur interne de position du fichier dont le descripteur est `$desc` **de** `$offset` **octets. Le paramètre** `$orig` **donne l'origine:** `SEEK_CUR` **(décalage à partir de la position courante),** `SEEK_SET` **(position absolue dans le fichier),** `SEEK_END` **(décalage à partir de la fin de fichier).**

Interface simplifiée

```
file_get_contents($nomfichier)
```

Ouvre un fichier dont on donne le nom et renvoie son contenu sous forme d'une chaine de caractères

```
file($nomfichier)
```

Renvoie un tableau avec une case par ligne dans le fichier. Chaque entrée contient le "\n" terminal.

Plan

1 Réseaux, TCP/IP ✔
2 Web et HTML ✔
3 CSS ✔
4 PHP : Introduction ✔
5 PHP : expressions régulières, fichiers, sessions
- 5.1 Manipulation des chaînes et expressions régulières ✔
- 5.2 Manipulation de fichiers ✔
- 5.3 En-tête de requêtes HTTP
- 5.4 Cookies
- 5.5 Sessions

Retour sur le protocole HTTP

- **Client :**

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

- **Serveur :**

```
HTTP/1.1 200 OK                          } ← code de retour
Server: nginx/1.4.1 (Ubuntu)
Date: Sun, 17 Nov 2013 16:44:48 GMT
Content-Type: text/html                  } ← type de contenu
Content-Length: 2038                     } ← longueur du contenu

<html>
  <head><title>Homepage</title> </head>
  <body>                                 } ← 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
<?php
  header('Content-type: application/csv');
  header('Content-Disposition: attachement; 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; filname="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 intervale 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 plusieur 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)

⇒ difficle de réaliser une « application » moderne répartie sur plusieurs pages

Plan

1 Réseaux, TCP/IP ✔
2 Web et HTML ✔
3 CSS ✔
4 PHP : Introduction ✔
5 PHP : expressions régulières, fichiers, sessions
- 5.1 Manipulation des chaînes et expressions régulières ✔
- 5.2 Manipulation de fichiers ✔
- 5.3 En-tête de requêtes HTTP ✔
- 5.4 Cookies
- 5.5 Sessions

Cookies

Un ***cookie*** est un paquet 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 caraceres
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* (1er 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);
```

Petite digression sur «*epoch*»

Représenter le temps (une date) dans un programme informatique est quelque chose de compliqué. Quels problèmes cela pose-t-il ?

- Fuseaux horaires
- Conversion de temps (de fuseau)
- Taille des entiers (Bug de l'an 2000, de l'an 2038, de l'an 292 277 026 596)
- Secondes intercalaires (« *leap second* »)

Ce n'est pas encore quelque chose de bien maîtrisé !

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 Réseaux, TCP/IP ✔
2 Web et HTML ✔
3 CSS ✔
4 PHP : Introduction ✔
5 PHP : expressions régulières, fichiers, sessions
  5.1 Manipulation des chaînes et expressions régulières ✔
  5.2 Manipulation de fichiers ✔
  5.3 En-tête de requêtes HTTP ✔
  5.4 Cookies ✔
  5.5 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 boutton 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 souhaîte 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 temp 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)

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

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

Avantages et inconvénients des session

+ 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 (envoie du cookie ("php_ssid", "12345"))

Coté serveur (PHP)

`session_start();`

- génération d'un ID unique "12345"
- dépot 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**