

# Cours Web n°8

## PHP — Notions avancées

Sandrine-Dominique Gouraud (gouraud@lri.fr)  
Pierre Senellart (pierre@senellart.com)



Semaine du 20 novembre 2006

# Plan du cours

- 1 Gestion des fichiers
- 2 Expressions régulières
- 3 Authentification, Sessions
- 4 Références
- 5 Application

- Les informations relatives aux fichiers transférés sont disponibles dans un tableau associatif **\$\_FILES** :
  - ▶ les clés sont les noms des champs de formulaire d'où provient le fichier
  - ▶ les valeurs sont des ensembles de propriétés (décrits comme des tableaux associatifs) décrivant le fichier reçu par le serveur auxquelles s'ajoute la propriété *error* qui permet de savoir si le transfert s'est bien déroulé

### Exemple (dans un fichier *FormTransfert.html*)

```
<form enctype="multipart/form-data"
  action="TransfertFichier.php" method="post">
  ...
  <div>
    <label for="maPhoto">Choisissez un fichier :</label>
    <input type="file" name="maPhoto" id="maPhoto" />
  </div>
  ...
</form>
```

`name` est le nom du fichier sur la machine du client

`tmp_name` est le nom du fichier temporaire sur la machine du serveur

`size` est la taille du fichier, en octets

`type` est le type MIME du fichier, par exemple "image/gif"

### Exemple (dans le fichier *TransfertFichier.php*)

```
...
$fichier=$_FILES['maPhoto'];
echo "Nom fichier client:".$fichier['name']."<br />";
echo "Nom fichier serveur:".$fichier['tmp_name']."<br />";
echo "Taille du fichier:".$fichier['size']."<br />";
echo "Type du fichier:".$fichier['type']."<br />";
...
```

`UPLOAD_ERR_OK` pas d'erreur, le transfert s'est bien passé

`UPLOAD_ERR_INI_SIZE` le fichier transmis dépasse la taille maximale autorisée

`UPLOAD_ERR_PARTIAL` le fichier est transféré seulement partiellement

`UPLOAD_ERR_NO_FILE` aucun fichier n'a été transféré

### Exemple

```
...  
$codeErreur= $_FILES['maPhoto']['error'];  
...
```

- La fonction PHP `copy(source,destination)` permet de copier le fichier *source* vers *destination*. Important parce que le fichier temporaire pourra être détruit à la fin du script!
  - ▶ **Attention** : le programme doit avoir les droits d'accès et d'écriture sur les répertoires dans lesquels les fichiers sont copiés
- La fonction `md5(ch)` permet de générer une nouvelle chaîne de caractères à partir de *ch*. Il est à peu près impossible d'obtenir deux valeurs identiques pour des chaînes différentes ce qui permet de considérer cette fonction comme un cryptage de chaîne. On peut en particulier utiliser md5 pour générer un nom de fichier quand on n'a rien de mieux (par exemple, un identifiant).

## Exemple

```
...  
// Copie du fichier dans le répertoire PHOTOS  
copy($fichier['tmp_name'], "./PHOTOS/$id.jpg");  
...
```

# Plan du cours

- 1 Gestion des fichiers
- 2 Expressions régulières**
- 3 Authentification, Sessions
- 4 Références
- 5 Application

- Les expressions régulières permettent de définir des *motifs* que l'on peut ensuite rechercher dans une chaîne de caractères.
- Une expression décrit un motif en indiquant :
  - ▶ le caractère ou la sous-chaîne attendu
  - ▶ l'ordre des caractères et des sous-chaînes
  - ▶ le nombre d'occurrences de ces caractères ou des ces sous-chaînes
- Le motif le plus simple est la sous-chaîne constante

## Exemple

L'expression régulière **foo** représente la sous-chaîne *foo*.

(motif) même chose que *motif*

$\wedge$ motif représente toutes les chaînes commençant par *motif*

motif\$ représente toutes les chaînes terminant par *motif*

$m^*$  indique que le motif *m* peut être présent 0 ou plusieurs fois

$m^+$  indique que le motif *m* peut être présent 1 ou plusieurs fois

$m?$  indique que le motif *m* peut être présent 0 ou 1 fois

$m\{a,b\}$  indique que le motif *m* peut être présent au moins *a* fois et au plus *b* fois

$m\{a,\}$  indique que le motif *m* peut être présent au moins *a* fois mais sans maximum

$m\{p\}$  est équivalent à  $m\{p,p\}$

## Exemple

	momomoouf	cmooopoue
<code>(mo){3}</code>	oui	non
<code>mo{3}</code>	non	oui
<code>a?</code>	oui	oui
<code>b*</code>	oui	oui
<code>p+</code>	non	oui
<code>o{4,5}</code>	non	non
<code>^cmo</code>	non	oui
<code>f\$</code>	oui	non

- [motif] toutes les chaînes avec un *m*, un *o*, un *t*, un *i* ou un *f*
- [a-f] toutes les chaînes avec une lettre entre *a* et *f*
- [a-zA-Z] toutes les chaînes avec une lettre de l'alphabet
- [^0-9] toutes les chaînes sans chiffre
- . représente n'importe quel caractère

**Remarque :** les caractères spéciaux `^`, `.`, `[`, `]`, `(`, `)`, `*`, `+`, `?`, `{`, `}` et `\` doivent être précédés par un `\` pour être pris en compte de manière littérale dans une expression régulière

### Exemple

`( \ ( [^\)]* \ ) )*` représente les chaînes de caractères bien parenthésées (avec un seul niveau, pas d'imbrication !).

`[:alpha:]` représente n'importe quel caractère alphanumérique

`[:blank:]` représente un espace ou une tabulation

`[:lower:]` représente une minuscule

`[:upper:]` représente une majuscule

`[:space:]` représente un espace, une tabulation ou un retour à la ligne

## Exemple

`[[:upper:]]0-9` représente un caractère quelconque parmi l'ensemble des lettres majuscules et des chiffres de 0 à 9.

`ereg(m,ch)` retourne vrai si le motif *m* a été trouvé dans la chaîne *ch*

`ereg(m,ch,tab)` retourne vrai si le motif *m* a été trouvé dans la chaîne *ch* et stocke dans le tableau *tab* toutes les occurrences trouvées dans *ch* du motif *m*

`ereg_replace(m,r,ch)` retourne la chaîne *ch* dans laquelle les occurrences du motif *m* ont été remplacées par la sous-chaîne *r*

## Exemple

```
if (ereg("<[^>]*>",$film['nomRealisateur'],$balises))
    $mes= "Le nom contient la balise:"
        . htmlspecialchars($balises[0]);
if (ereg("[^a-zA-Z]",$film['nomRealisateur']))
    $mes= "Le nom contient un ou plusieurs"
        . "caractères non-alphabétiques : "
        . ereg_replace("[^a-zA-Z]","*",$film['nomRealisateur']);
```

# Plan du cours

- 1 Gestion des fichiers
- 2 Expressions régulières
- 3 Authentification, Sessions**
- 4 Références
- 5 Application

**session** : ensemble d'informations conservées tout au long d'une interaction avec les différentes pages d'un site Web

**authentification** : mécanisme permettant d'associer un **identifiant** (login) à un utilisateur d'un site Web, de manière à permettre de la personnalisation du contenu, ou de la gestion de droits sur une application Web ; habituellement, l'authentification est faite grâce à un **mot de passe**.

- Méthode d'authentification HTTP simple disponible au niveau du serveur Web, mais impose une modification de la configuration de celui-ci ; un peu lourd, difficilement connectable à un SGBD.
- Pas de gestion de session à proprement parler en HTTP. Alternatives :
  - ▶ Paramètres HTTP cachés dans l'URL (méthode GET). Mécanisme un peu lourd, puisque tous les liens doivent être changés pour incorporer ces paramètres.
  - ▶ Cookies.

- Informations, sous la forme de clés/valeurs, qu'un serveur Web demande à un client Web de conserver et de retransmettre à chaque requête HTTP.
- `setcookie($name,$value)` : demande client de stocker un cookie de nom `$name` et de valeur `$value`.
- `$_COOKIE` est un tableau associatif des cookies que le client a envoyé au serveur Web.

PHP fournit une abstraction de la gestion de session (utilisant un cookie, mais sans avoir à le gérer soi-même).

`session_start()` ouvre une session en cours, ou crée une nouvelle session s'il n'y a pas de session ouverte; ceci est à placer **au tout début du script PHP**, ou en tous cas avant que quoi que ce soit n'ait été écrit dans la page, de manière à pouvoir modifier les en-têtes de la réponse HTTP.

`session_destroy()` termine la session en cours.

`session_id()` fournit un identifiant de la session en cours.

`$_SESSION` contient l'ensemble des paramètres de session (tableau associatif clé/valeur), disponibles dans les différentes pages Web de la même session.

- Un formulaire demande login et mot de passe.
- Un script de traitement de ce formulaire, contrôle que le login et le mot de passe sont corrects (par exemple à l'aide d'une table MySQL) :
  - ▶ Si c'est le cas, crée une session PHP (`session_start();`), y ajoute un paramètre nommé par exemple `valid_user` (`$_SESSION['valid_user']=1;`) et redirige vers une autre page.
  - ▶ Sinon, redirige vers la page de formulaire.
- Les autres pages (pages auxquelles les utilisateurs authentifiés et seulement eux ont accès) commencent par un `session_start();` et contrôlent si l'utilisateur est identifié (`if($_SESSION['valid_user']==1;) { ...}`) et sinon redirigent vers la page de formulaire
- Une page de déconnexion appelle `session_destroy();`

# Plan du cours

- 1 Gestion des fichiers
- 2 Expressions régulières
- 3 Authentification, Sessions
- 4 Références**
- 5 Application

- <http://www.php.net/>
- *Pratique de MySQL et PHP*, Philippe Rigaux, O'Reilly
- *Maîtrise des expressions régulières*, O'Reilly
- *Requests for Comments* concernant la gestion des cookies :
  - ▶ <http://www.ietf.org/rfc/rfc2109.txt>
  - ▶ <http://www.ietf.org/rfc/rfc2965.txt>

# Plan du cours

- 1 Gestion des fichiers
- 2 Expressions régulières
- 3 Authentification, Sessions
- 4 Références
- 5 Application**

- 1 Écrire un script vérifiant (à l'aide d'une expression régulière) qu'un paramètre HTTP de nom *email* ressemble bien à un e-mail (le script devra afficher un message d'erreur si *email* vaut zorglub ou zkds@qdsj, mais pas si *email* vaut toto@titi.com ou gabou@areuh.fr)
- 2 Reproduire le comportement des scripts de démonstration login.php, bonjour.php et quitter.php à l'aide d'une session PHP :
  - 1 login.php présente un formulaire d'ouverture de session et redirige vers bonjour.php.
  - 2 bonjour.php stocke le nom d'utilisateur dans les paramètres de session et affiche un message contenant ce nom si la session est ouverte, affiche un autre message sinon.
  - 3 quitter.php affiche un message contenant le nom d'utilisateur et termine la session si la session est ouverte, affiche un autre message sinon.