Programmation Internet

Cours 7

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

Plan

Fonctions

Les fonctions sont déclarées à l'aide du mot-clé `function`. On renvoie des résultats à l'aide du mot-clé `return`.

```php
<?php
 function double ($x)
   {
      return $x + $x;
   }

   echo double(10);
?>
```

Portée des variables locales et globales ⚠

```php
<?php
$a = 42;
function add_a($x)
{

  return $x + $a;
}
echo add_a(10);
?>
```

```php
<?php
$a = 42;
function add_a($x)
{
  global $a;
  return $x + $a;
}
echo add_a(10);
?>
```

Le code de gauche affiche 10 ! Les variables ont une portée locale par défaut. Si `$a` n'est pas définie dans le corps de la fonction, sa valeur est `NULL` (variable non définie). Pour référencer des variables globales, on utilise le mot clé `global`.

Fonction : définition ⚠

On peut utiliser une fonction « avant » de la définir :

```
<?php
      echo next(10);
      echo <br/>;

      function next($x)
      {
        return $x + 1;
      }
?>
```

On définira toujours des fonctions avant de les utiliser. On ne peut pas définir deux fonctions avec le même nom.

Fonctions : passage par référence

On utiliser le modificateur & devant un paramètre de fonction pour indiquer que ce dernier est passé par référence.

```php
<?php
  function add_a($tab)
  {
    $tab["a"] = 42;
  }

  $mytab = array();
  add_a($mytab);
  echo $mytab["a"];
  //n'affiche rien (car NULL ⇝ "")
?>
```

```php
<?php
  function add_a(&$tab)
  {
    $tab["a"] = 42;
  }

  $mytab = array();
  add_a($mytab);
  echo $mytab["a"];
```

Plan

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)` : **converti 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...)+/', $chaine); // affiche 1
    echo preg_match('/[^A-Z]+/');             // affiche 0
    echo preg_match('/[^A-Z]*/');             // affiche 1 !
    echo preg_match('/^[^A-Z]*$/');           // 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
        $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
    $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

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.