TP4 : Rendre un site dynamique avec PHP

1 Objectifs

Utiliser le langage PHP pour produire des pages HTML.
Le but est de faire évoluer le site statique conçu lors du TP précédent en produisant les pages de façon modulaire en PHP.

2 Moyens nécessaires

- l'IDE Eclipse ;
- le navigateur Firefox ;
- le serveur HTTP Apache avec le support du langage PHP ;
- Le site statique précédemment réalisé s'il est au point ou le corrigé fourni.

3 Configuration du poste de travail

Pour faire fonctionner un site Web dynamique, vous aurez besoin :
D'un serveur HTTP (Web), en l'occurrence Apache ;
D'un interpréteur PHP couplé au serveur HTTP.

3.1 Test de fonctionnement du serveur

Question 1) Vérifiez que le serveur est en fonctionnement en entrant l'URL *localhost* dans la barre de navigation de Firefox.

Il faut aussi vérifier que le module PHP est bien utilisable avec le serveur.
Pour cela, on peut créer un fichier PHP *test.php* qui se contente d'appeler la procédure **phpinfo( )**.
Cette procédure crée et renvoie automatiquement une page donnant des informations très utiles sur l'installation de PHP dans le serveur Web.

Question 2) Créez le fichier de test dans un nouveau projet PHP nommé par exemple *Bloc1_TP4*. Essayez d'exécuter ce fichier de test depuis votre navigateur. Concluez.

3.2 Exécution du code PHP par Apache

Le code PHP doit être exécuté par le serveur Apache pour être transformé en HTML lisible par votre navigateur.
Le site Web de votre machine est hébergé dans le répertoire */var/www/html*.
En phase de développement, cela pose deux problèmes :

- Ce répertoire n'est accessible en écriture que par le compte d'administration *root* ;
- Votre espace "naturel" de développement est le "*workspace*" d'Eclipse.

La solution la plus simple est donc de créer un lien dans */var/www/html* qui pointe vers votre espace

de développement */SERVSMB/home/votre_nom/workspace*. La technique consiste à créer un lien symbolique.

> Qu'est-ce qu'un lien symbolique ? C'est un fichier spécial, sous Linux/Unix, qui pointe vers un autre fichier ou répertoire. Accéder au lien symbolique revient donc à accéder à la ressource vers laquel il pointe.
> Comment créer des liens symboliques ? Depuis la ligne de commande :
> (compte root sous Debian) **ln -s répertoire_cible nom_du_lien_créé**

Question 3) Utilisez cette méthode pour créer un lien de votre "*workspace*" vers */var/www/html/votre_nom*.

Question 4) Testez ensuite le fonctionnement en saisissant l'URL *http://localhost/votre_nom* dans le navigateur Web.

4 Evolution du site vers un site dynamique

La page d'accueil du site développé lors du TP précédent présentait différentes parties que l'on retrouve fréquemment dans beaucoup de sites :

- un bandeau de titre ;
- un menu ;
- une zone de texte principale ;
- un bandeau latéral d'informations ;
- un pied de page.

On souhaite une uniformité du site, quelle que soit la page visualisée. On choisit donc que certaines zones de la page seront présentes quelle que soit la page parcourue. Ces zones sont visibles sur la maquette suivante :

Le tout est complété d'une image d'arrière plan.

Le site statique a été modifié afin de facilité la maintenance (externalisation des éléments réutilisés dans plusieurs pages web). L'entête, le footer, le menu, le bloc infos et le bloc article ont été externalisés dans différents fichiers : *bloc_titre.html*, *bloc_pied.html*, *bloc_menu.html*, *bloc_infos.html*, *article_accueil.html*.

La nouvelle page d'accueil est composée d'un fichier PHP nommé *index.php* en remplacement du fichier *index.html*.

Ce fichier, on va appeler en PHP les différentes portions de code HTML qui vont composer la page. L'appel d'un code extérieur se fait à l'aide de la fonction **include 'nom_fichier'** de PHP.

De la même façon, la page "galerie de photos" a été reconstituée en utilisant la même structure.

4.1 Création dynamique de la zone d'informations avec PHP

Sur la page d'accueil, on peut visualiser les informations qui sont listées dans des cadres situés dans la zone d'informations, à droite de l'écran : le bloc infos.

On souhaite que cette zone d'information soit désormais construite automatiquement à partir des données fournies dans le répertoire *infos*.

Pour chaque information, on y trouve :

- Le fichier nommé *titre_info.txt* contenant le texte de l'information ;
- La photo illustrant l'information, *titre_info.jpg ;*
- Le titre de l'information à générer automatiquement sera donc donnée par le nom des fichiers textes utilisés dans le répertoire infos.

Chaque cadre d'information doit contenir :

- Le titre de l'information (le nom du fichier), de type titre h3 ;
- Quelques lignes de l'information (les lignes contenues dans le fichier txt). Le cadre contenant le texte a une hauteur limitée (200pixels) de façon à pouvoir visualiser plusieurs informations à la fois.
- Le bas du cadre comporte un lien hypertexte "lire la suite..." qui renvoie vers l'article complet. Le lien doit appeler l'URL *article_infos.php?info=titre_info* (*titre_info* est le titre de l'information à afficher). On traitera ultérieurement la page article_infos.

Question 5) Produisez le code PHP nécessaire dans le fichier *bloc_infos.php* pour afficher automatiquement le cadre d'infos et les différentes infos sur la page d'accueil.

Procédez par étapes (contenu du bloc, puis titre de l'article puis lien) afin de ne pas avoir trop d'erreurs en même temps.

Ce fichier *bloc_infos.php* remplacera le fichier statique *bloc_infos.html*.

Vous aurez besoin des fonctions **glob()**, et **basename() (voir documentation en annexe)**.

4.2 Création dynamique d'un article d'information

Le lien hypertexte en fin de bloc information renvoie vers l'article complet avec sa photo d'illustration, par un appel de l'URL : article_*info.php?info=titre_info*

La maquette d'un article sera la suivante :

L'article va être composé dynamiquement à partir du fichier *info.php* appelé avec le paramètre *info*. Le passage de paramètre est fait en utilisant la méthode HTTP GET.

Récupération de paramètres par le code PHP :

Les paramètres passés par la méthode GET sont automatiquement placés dans le tableau associatif $_GET. Ainsi, on peut récupérer la valeur du paramètre info en lisant $_GET['info']

On va donc pouvoir récupérer le nom de l'article dans le tableau $_GET['info'] puis on pourra ensuite reconstituer l'article avec les éléments contenus dans le fichier texte et avec l'image du même nom.

Question 6) Produisez le code PHP nécessaire dans le fichier *info.php* pour obtenir l'affichage automatique d'un article dont la référence est passée en paramètre par la méthode GET.

5 Pour les plus rapides

5.1 Amélioration de l'affichage des informations

Dans le code produit précédemment, le titre d'une information est obtenu à partir du nom du fichier la contenant. Cela est une limitation sérieuse.
On propose de modifier le code de façon à ce que :

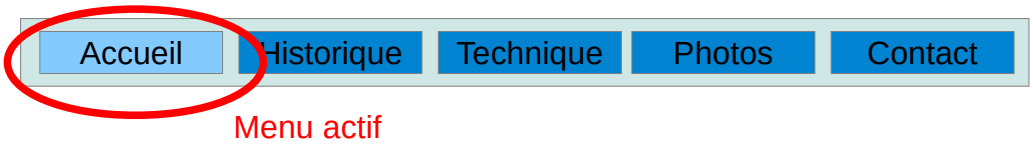
- Le titre soit fourni par la première ligne du fichier ;

- La date de parution soit fournie par la deuxième ligne du fichier
- Chaque retour à la ligne dans le fichier texte donne lieu à un retour à la ligne dans l'affichage produit.

Question 7) Modifiez le code de façon à répondre aux nouvelles exigences. Vous devrez utiliser les fonctions **file_get_contents( )** et **explode()**.

5.2 Gestion du menu

Le menu doit être presque identique d'une page à l'autre, en dehors du fait que la page active y est visible.



Menu actif

Ce menu doit donc être créé dynamiquement en PHP. Plusieurs solutions sont possibles.

Question 8) Produisez le code qui générera le menu dans le fichier *bloc_menu.php* remplaçant le fichier *bloc_menu.html* et testez-le résultat en fonction des pages affichées.

6 Annexes

6.1 Fonction Glob

glob

(PHP 4 >= 4.3.0, PHP 5)

glob — Recherche des chemins qui vérifient un masque

Description

```
array glob ( string $pattern [, int $flags = 0 ] )
```

glob() recherche tous les chemins qui vérifient le masque *pattern*, en suivant les règles utilisées par la fonction *glob()* de la libc, qui sont les mêmes que celles utilisées par le Shell en général.

Liste de paramètres

pattern

Le masque. Aucun remplacement de tilde (~) ou de paramètre n'est fait.

flags

Les drapeaux valides sont :

- **GLOB_MARK** : Ajoute un slash final à chaque dossier retourné
- **GLOB_NOSORT** : Retourne les fichiers dans l'ordre d'apparence (pas de tri)
- **GLOB_NOCHECK** : Retourne le masque de recherche si aucun fichier n'a été trouvé
- **GLOB_NOESCAPE** : Ne protège aucun métacaractère d'un antislash
- **GLOB_BRACE** : Remplace {a,b,c} par 'a', 'b' ou 'c'
- **GLOB_ONLYDIR** : Ne retourne que les dossiers qui vérifient le masque
- **GLOB_ERR** : Stop lors d'une erreur (comme des dossiers non lisibles), par défaut, les erreurs sont ignorées.

Valeurs de retour

Retourne un tableau contenant les fichiers et dossiers correspondant au masque, un tableau vide s'il n'y a aucune correspondance, ou **FALSE** si une erreur survient.

Note:

Sous certains systèmes, il est impossible de distinguer un masque vide d'une erreur.

Historique

Version	Description
5.1.0	**GLOB_ERR** a été ajouté
4.3.3	**GLOB_ONLYDIR** devient disponible sous Windows ainsi que sur les autres systèmes n'utilisant pas la bibliothèque GNU C.

Exemples

Exemple #1 Un moyen pratique pour remplacer opendir() par glob()

```php
<?php
foreach (glob("*.txt") as $filename) {
        echo "$filename occupe " . filesize($filename) . "\n";
}
?>
```

6.2 Fonction Basedir

basename

(PHP 4, PHP 5)

basename — Retourne le nom du fichier dans un chemin

Description

```
string basename ( string $path [, string $suffix ] )
```

Prend en paramètre *path*, le chemin complet d'un fichier et en extrait le nom du fichier.

Liste de paramètres

path

Un chemin.

Sous Windows, les caractères (/) et antislash (\) sont utilisés comme séparateurs de dossier. Sous les autres OS, seul le caractère slash (/) est utilisé.

suffix

Si *suffix* est fourni, le suffixe sera aussi supprimé.

Valeurs de retour

Retourne le nom de base du chemin *path*.

Historique

Version	Description
4.1.0	Le paramètre *suffix* a été ajouté.

Exemples

Exemple #1 Exemple avec basename()

```
1. <?php
2. echo "1) ".basename("/etc/sudoers.d", ".d").PHP_EOL;
3. echo "2) ".basename("/etc/passwd").PHP_EOL;
4. echo "3) ".basename("/etc/").PHP_EOL;
5. echo "4) ".basename(".").PHP_EOL;
6. echo "5) ".basename("/");
7. ?>
```

L'exemple ci-dessus va afficher :

```
1) sudoers
2) passwd
3) etc
4) .
5)
```

6.3 Fonction File_get_contents

file_get_contents

(PHP 4 >= 4.3.0, PHP 5)

file_get_contents — Lit tout un fichier dans une chaîne

Description

```
string file_get_contents ( string $filename [, bool $use_include_path = false [, resource $context [, int $offset = -1 [, int $maxlen ]]]] )
```

Identique à la fonction file(), hormis le fait que **file_get_contents()** retourne le fichier *filename* dans une chaîne, à partir de la position *offset*, et jusqu'à *maxlen* octets. En cas d'erreur, **file_get_contents()** retourne **FALSE**.

file_get_contents() est la façon recommandée pour lire le contenu d'un fichier dans une chaîne de caractères. Elle utilisera un buffer en mémoire si ce mécanisme est supporté par votre système, afin d'améliorer les performances.

Note:

Si vous ouvrez une URI avec des caractères spéciaux, comme des espaces, vous devez encoder cette URI avec la fonction urlencode().

Liste de paramètres

filename

Nom du fichier à lire.

use_include_path

Note:

Depuis PHP 5, la constante **FILE_USE_INCLUDE_PATH** peut être utilisée pour déclencher la recherche dans le chemin d'inclusion.

context

Une ressource de contexte valide, créée avec la fonction stream_context_create(). Si vous n'avez pas besoin d'utiliser un contexte particulier, vous pouvez ignorer ce paramètre en affectant la valeur **NULL**.

offset

La position à partir de laquelle on commence à lire dans le flux original.

Le déplacement dans le fichier (*offset*) n'est pas supporté sur des fichiers distants. Si vous tentez de vous déplacer dans un fichier qui n'est pas un fichier local peut fonctionner sur les petits déplacements, mais le comportement peut ne pas être attendu car le processus utilise le flux du buffer.

maxlen

La taille maximale des données à lire. Le comportement par défaut est de lire jusqu'à la fin du fichier. Ce paramètre s'applique sur le flux traité par les filtres.

Valeurs de retour

Retourne les données lues ou **FALSE** si une erreur survient.

Avertissement

Cette fonction peut retourner **FALSE**, mais elle peut aussi retourner une valeur équivalent à **FALSE**. Veuillez lire la section sur les booléens pour plus d'informations. Utilisez l'opérateur === pour tester la valeur de retour exacte de cette fonction.

Erreurs / Exceptions

Émets une alerte de type **E_WARNING** si, *filename* ne peut être trouvé, si le paramètre *maxlength* est plus petit que zéro, ou si le déplacement à la position *offset* spécifié dans le flux échoue.

Exemples

Exemple #1 Lit et affiche le code HTML d'un site Web

```php
<?php
$homepage = file_get_contents('http://www.example.com/');
echo $homepage;
?>
```

Exemple #2 Recherche un fichier dans le include_path

```php
<?php
// avant PHP 5
$file = file_get_contents('./people.txt', true);
// depuis PHP 5
$file = file_get_contents('./people.txt', FILE_USE_INCLUDE_PATH);
?>
```

6.4 Fonction Explode

explode

(PHP 4, PHP 5)

explode — Coupe une chaîne en segments

Description

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

explode() retourne un tableau de chaînes, chacune d'elle étant une sous-chaîne du paramètre *string* extraite en utilisant le séparateur *delimiter*.

Liste de paramètres

delimiter

Le séparateur.

string

La chaîne initiale.

limit

Si *limit* est défini et positif, le tableau retourné contient, au maximum, *limit* éléments, et le dernier élément contiendra le reste de la chaîne.

Si le paramètre *limit* est négatif, tous les éléments, excepté les -*limit* derniers éléments sont retournés.

Si *limit* vaut zéro, il est traité comme valant 1.

Bien que implode() puisse, pour des raisons historiques, accepter ces paramètres dans n'importe quel ordre, **explode()** ne le peut pas. Vous devez vous assurer que le paramètre *delimiter* soit placé avant le paramètre *string*.

Valeurs de retour

Retourne un tableau de chaînes de caractères créées en découpant la chaîne du paramètre *string* en plusieurs morceaux suivant le paramètre *delimiter*.

Si *delimiter* est une chaîne vide (""), **explode()** retournera **FALSE**. Si *delimiter* contient une valeur qui n'est pas contenue dans *string* ainsi qu'une valeur négative pour le paramètre *limit*, alors **explode()** retournera un tableau vide, sinon, un tableau contenant la chaîne *string* entière.

Historique

Version	Description
5.1.0	Le paramètre *limit* peut désormais être négatif
4.0.1	Le paramètre *limit* a été ajouté

Exemples

Exemple #1 Exemple avec explode()

```php
<?php
// Exemple 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2

// Exemple 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user, $pass, $uid, $gid, $gecos, $home, $shell) = explode(":", $data);
echo $user; // foo
echo $pass; // *

?>
```