Formations en Informatique de Lille – 2016/2017

Feuille 2 - Technologies du Web

janvier 2017

PHP

Exercice 1 : Préparation de l'espace de travail.

- Sur webtp créez un dossier `feuille2` consacré au travail de cette semaine
- Dans ce dossier `feuille2` , transférez les fichiers de l'archive, y compris ses sous-dossiers et leur contenus.
  - `couvertures` contient des images
  - `data` contient les données qui serviront à engendrer la page HTML
  - `lib` est detsiné à un ou plusieurs fichiers PHP qui n'ont pas vocation à être exécutés directement mais à être inclus dans d'autres.

  N'oubliez pas de mettre les droits convenables sur l'ensemble des dossiers et fichiers.
- **Les fonctions que vous écrirez dans le cadre de cet exercice figureront dans le fichier** `lib/fonctionsLivre.php`

Le but final de l'exercice sera de générer en PHP un document HTML à partir des données figurant dans le fichier `exempleLivre.txt`. Nous procèderons par étapes, en structurant le code PHP.

Le fichier `exempleLivre.txt` contient la description d'un ouvrage. Examinez son contenu. Le livre y est décrit par une suite de lignes, chacune dédiée à un type d'information particulier.
Chaque ligne se compose de 2 parties, séparées par le premier caractère `:`

- avant le premier `:`, le **nom de la propriété**
- après le premier `:`, la **valeur de la propriété**

Les espaces éventuels situés au début ou en fin de ligne, ou à côté du séparateur `:` ne sont pas significatifs (ils ne font partie ni du nom ni de la valeur de la propriété).
La description d'un livre est une suite de lignes de propriété. Elle est suivie soit d'une ligne vide, soit de la fin de fichier

Question 1.1 : *Décomposition d'une ligne de propriété.*

Pour cette question, vous consulterez la documentation des fonctions PHP `strpos, trim`
Écrivez une fonction `propertyName($line)` dont l'argument est une chaîne (description de propriété) et dont le résultat est une chaîne contenant uniquement le **nom** de la propriété (sans espaces ni avant ni après le nom). Si la ligne ne comporte pas le caractère `:`, une exception sera déclenchée.
Réalisez de même une fonction `propertyValue($line)`

Test unitaire des fonctions que vous venez d'écrire
La base d'un fichier `debug.php` vous est fournie. Sa première instruction :

```
header('Content-Type: text/plain;charset=UTF-8');
```

indique au navigateur que le script va produire du texte ordinaire et non de l'HTML. Le texte produit sera donc affiché tel quel par le navigateur, sans mise en page ni transformation.
La deuxième instruction :

```
require_once('lib/fonctionsLivre.php');
```

permet d'**inclure** le fichier où se trouvent les fonctions.
Les lignes suivantes permettent de faire des tests, vous pouvez les compléter éventuellement.

Avant d'aller plus loin, vérifiez la validité de vos 2 fonctions.

Question 1.2 : *Lecture des données et représentation en PHP*

Un livre sera représenté par une table PHP associative dont les clés sont les noms de propriétés. À chaque clé est associée la valeur correspondante. Par exemple :

clés	valeurs
`couverture`	`scorpion.jpg`
`titre`	`La marque du diable`
`serie`	`Le Scorpion`
`auteurs`	`Marini - Desberg`
`année`	`2000`
`catégorie`	`bandes-dessinées`

Construire une fonction PHP `readBook($file)` qui reçoit en argument un descripteur de fichier (supposé déja ouvert). Le fichier est censé contenir la description d'un livre. Le résultat de la fonction est une table PHP représentant ce livre.
On rappelle que la description d'un livre est suivie soit d'une ligne vide, soit de la fin de fichier.

Test unitaire des fonctions que vous venez d'écrire
Reprenez le script de débogage, mettez en commentaire les tests précédents et activez ceux de cette question.
La fonction `print_r()` permet d'afficher le contenu d'une variable quelconque, y compris tableau ou objet. **Elle ne doit être utilisée QUE pour du débogage.**

Question 1.3 : *Construction du code HTML à partir des données.*
Le modèle de document à produire vous est donné dans le fichier `exempleLivre.html`. Examinez-en le contenu. Vous remarquerez que chaque propriété se traduit par un élément HTML (pas toujours le même : `div`, `time`, `h2` ...)

- cet élément possède un attribut `class`. La classe de l'élément est le nom de la propriété.
- le contenu de l'élément est en général la valeur de la propriété. Dans certains cas cette valeur a été transformée (pour les auteurs ou la couverture).

Vous allez maintenant construire les fonctions nécessaires à la production du code HTML en procédant de façon modulaire.
→ Notez bien que chacune des fonctions que vous allez écrire maintenant **ne doit rien envoyer sur la sortie standard** (elles ne comportent aucun echo, aucun printf...). Seul le **résultat** de chaque fonction est important.

1. Construire une fonction `elementBuilder($propName, $elementType, $text)` dont le résultat est une chaîne contenant le code HTML d'un élément de type `$elementType`, de classe `propertyName` et de contenu `$text`.

   Par exemple `elementBuilder('titre', 'h2', 'La marque du diable')` doit renvoyer la chaîne `'<h2 class="titre">La marque du diable</h2>'`

   NB : longueur approximative : 1 ligne environ

   Procédez au test unitaire de la fonction, en adaptant le script de débogage.

2. Construire une fonction `authorsToHTML($authors)` qui renvoie une chaîne contenant le code HTML représentant les auteurs. Dans `authors`, les noms des auteurs sont séparés par `' - '`. Dans le résultat, chaque nom d'auteur doit être inséré dans un élément `span`. Les spans sont séparés par un espace.

   par exemple `authorsToHTML('Marini - Desberg')` a pour résultat la chaîne

   `'<span>Marini</span> <span>Desberg</span>'`

   NB : vous utiliserez les fonctions `explode()` et `implode()` (doc à consulter)

   NB : longueur approximative : 2 lignes environ

   Procédez au test unitaire de la fonction, en adaptant le script de débogage.

3. Construire une fonction `coverToHTML($fileName)` qui renvoie une chaîne contenant le code HTML représentant l'image de couverture, c'est à dire un élément `img` dont l'attribut `src` renvoie à l'URL du fichier, dans le dossier `couvertures` et dont l'attribut `alt` est défini. Par exemple `coverToHTML('scorpion.jpg')` a pour résultat la chaîne

   `'<img src="couvertures/scorpion.jpg" alt="image de couverture" />'`

NB : longueur approximative : 1 ligne environ

Procédez au test unitaire de la fonction, en adaptant le script de débogage.

4. Construire une fonction `propertyToHTML($propName,$propValue)` qui renvoie une chaîne contenant le code HTML représentant la propriété, selon le principe :

propriété	type d'élément	contenu de l'élément
titre	h2	valeur (telle quelle)
couverture	div	élément `img`
auteurs	div	suite d'éléments `span`
année	time	valeur (telle quelle)
autres propriétés	div	valeur (telle quelle)

Par exemple `propertyToHTML('titre', 'La marque du diable')` a pour résultat la chaîne

```
'<h2 class="titre">La marque du diable</h2>'
```

`propertyToHTML('auteurs', 'Marini - Desberg')` a pour résultat la chaîne

```
'<div class="auteurs"><span>Marini</span> <span>Desberg</span></div>'
```

NB : longueur approximative : 10 lignes environ

Procédez au test unitaire de la fonction, en adaptant le script de débogage.

5. Enfin, construisez une fonction `bookToHTML($book)` dont l'argument est une table PHP représentant un livre (voir question précédente) et dont le résultat est une chaîne qui contient le code HTML correspondant (c'est à dire le code d'un élément `article`).

   Pour la structure HTML à produire, vous vous réfèrerez au contenu de fichier `exempleLivre.html`. Vous remarquerez ainsi que l'ensemble des propriétés est regroupé dans un élément `div` de classe `description`, à l'exception de la propriété `couverture` qui figure en premier et n'est pas groupée avec les autres.

   NB : longueur approximative : 10 lignes environ

   Procédez au test unitaire de la fonction, en adaptant le script de débogage.

Question 1.4 : *Le travail final : la page PHP*
En utilisant les fonctions précédentes, concevez, directement dans le dossier `feuille2`, la page `livreUnique.php` qui produit à partir du fichier `exempleLivre.txt` un code HTML analogue à celui de la page `exempleLivre.html`

Exercice 2 :

Question 2.1 : Vous allez modifier le code de la fonction `readBook()` développée dans l'exercice précédent (conservez une copie de la version précédente que vous mettrez en commentaires).

On suppose maintenant que dans `$file`, à partir de la position courante, on peut éventuellement trouver une suite de lignes vides précédant la description d'un livre. Si toutefois `$file` ne comporte plus aucune ligne non vide, alors la fonction `readBook($file)` doit renvoyer `FALSE`.

Cela ne devrait pas impacter le fonctionnement du script de l'exercice précédant : vérifiez qu'il fonctionne toujours avec la nouvelle version de la fonction.

Question 2.2 : Créez une fonction `libraryToHTML($file)`. Son argument `$file` est supposé être un descripteur d'un fichier ouvert contenant une suite (éventuellement vide) de descriptions de livres. Un exemple de tel fichier est `livres.txt`
Le résultat de la fonction doit être une chaîne qui contient le code HTML présentant l'ensemble des livres du fichier (donc une suite d'articles).

Question 2.3 : Créez une page `bibliotheque.php` qui présente (en HTML) l'ensemble des livres figurant dans `livres.txt`

Question 2.4 : (à faire en dehors des séances) Associer aux pages `livreUnique.php` et `bibliotheque.php` un style permettant d'obtenir une meilleure présentation. Voir à ce sujet l'exercice du l'UE TW1 consacré à cette question.

Exercice 3 : *Sécurisation du site*

Dans l'état actuel, l'ensemble des fichiers est accessible directement depuis n'importe quel client du site web, comme vous allez le voir

— Dans le navigateur, entrez l'URL suivante :
`http://webtp.fil.univ-lille1.fr/~votre_login/feuille2/data/livres.txt`
Le navigateur affiche alors le fichier de données.

— Dans le navigateur, entrez l'URL suivante :
`http://webtp.fil.univ-lille1.fr/~votre_login/feuille2/lib/fonctionsLivre.php`
Le navigateur affiche une page blanche mais pas de message d'erreur : le script `fonctionsLivre.php` vient d'être exécuté, mais comme il ne produit pas de sorties, rien ne s'affiche.

Les deux manipulations ci-dessus mettent en évidence des problèmes.

— l'utilisateur a pu visualiser directement des données brutes qui ne lui étaient pas, en principe, destinées (le fichier `livres.txt`)

— l'utilisateur a déclenché l'exécution d'un script qui n'était pas prévu pour être exécuté directement mais pour être inclus dans un autre.

Un principe basique de sécurisation d'un site est que **les seuls fichiers accessibles directement sont les pages PHP, les fichiers nécessaires à la présentation de ces pages (CSS, JS), les media inclus dans ces pages (images).**

Ainsi l'organisateur du site ne doit PAS se demander « devrais-je interdire l'accès à tel fichier ? » mais « faut-il vraiment que je laisse libre accès à tel fichier ? »

Question 3.1 : *Protection des dossiers*

Créez un fichier nommé `.htaccess` contenant les lignes :

```
Order deny, allow
Deny from all
```

Il s'agit d'un fichier de directives pour le serveur web. Ici, la directive donnée est de ne pas laisser libre accès au contenu du dossier où se trouve le fichier `.htaccess`

Un fichier d'un dossier ainsi protégé n'est plus accessible directement depuis un client, mais il reste néanmoins accessible depuis les autres scripts : il peut toujours être inclus (par un `require`) ou manipulé en tant que fichier (`fopen, fgets` ...)

— Placez une copie de ce fichier dans le dossier `data` et une autre dans le dossier `lib`.

— Tentez d'accéder de nouveau aux deux fichiers qui étaient accessibles tout à l'heure

Notez que le dossier `couvertures` doit rester accessible : il contient des images faisant partie des pages web du site.

Notez aussi que le script `debug.php` doit être supprimé du site (ou déplacé dans un répertoire protégé) quand le site est en phase de production.