Formations en Informatique de Lille – 2016/2017

Semaine 4 - Technologies du Web

février 2017

PHP - SVG - Formulaires

Exercice 1 : Recopiez le fichier `testSVG.html` dans votre répertoire. Il s'agit d'un fichier HTML 5 qui comporte un dessin en SVG.
SVG est un langage XML consacré au dessin vectoriel (les figures sont décrites par leur caractéristiques géométriques).

Ouvrez le fichier dans un éditeur et examinez attentivement le contenu de l'élément d'identifiant `id=dessin` et les commentaires qui expliquent la signification des balises `circle`, `rectangle` et `polygon`.
Question 1.1 :

1. créez une bibliothèque `fonctionsSVG.php`. Écrivez une fonction `cercle($cx,$cy,$r)` qui renvoie une chaîne contenant le texte de l'élément SVG `<circle>` dessinant un cercle de centre (cx, cy) et de rayon r.
2. copiez le fichier `testSVG.html` en un fichier `figures.php` dans lequel vous remplacerez le contenu de l'élément d'identifiant `dessin` par un bloc PHP. Celui ci affichera, en utilisant la fonction ci-dessus, un cercle de centre $(150, 100)$ et de rayon 75. Testez.
3. modifiez `figures.php` pour qu'il prenne en compte les paramètres (obligatoires) suivants, passés en mode GET :
   — `cx` : un entier
   — `cy` : un entier
   — `r` : un entier positif
   Ces paramètres déterminent les caractéristiques du cercle à engendrer. Si l'un des paramètres est manquant ou incorrect, le script doit afficher la page d'erreur fournie au lieu de la page normale. (utiliser `require(...)` suivi de `exit()`)

Question 1.2 : Le carré possédant 2 côtés horizontaux, **inscrit** dans le cercle de centre (cx, cy) et de rayon r possède
— des côtés de longueur $c = r\sqrt{2}$.
— pour coin inférieur le point $(cx - \frac{c}{2}, cy - \frac{c}{2})$

1. écrivez une fonction `carre($cx,$cy,$r)` qui renvoie une chaîne contenant le texte de l'élément SVG `<rect>` dessinant un carré «horizontal» inscrit dans le cercle de centre (cx, cy) et de rayon r.
2. complétez le script `figures.php` pour qu'il dessine un cercle et le carré horizontal inscrit dans ce cercle.
3. ajoutez à la fonction `carre()`, un paramètre `angle` optionnel (valeur par défaut : 0) qui a pour effet de pivoter le carré autour de son centre.
4. modifiez le script `figures.pjp` pour prendre en compte un argument HTTP facultatif nommé `a` représentant l'angle de rotation

Question 1.3 : Le triangle équilatéral à base horizontale **inscrit** dans le cercle de centre (cx, cy) et de rayon r possède les 3 sommets $(cx, cy + r)$, $(cx - \frac{r\sqrt{3}}{2}, cy - \frac{r}{2})$, $(cx + \frac{r\sqrt{3}}{2}, cy - \frac{r}{2})$
Écrivez la fonction `triangleInscrit($cx,$cy,$r,$angle=0)` qui renvoie l'élément SVG `<polygon>` dessinant un triangle équilatéral inscrit dans le cercle de centre (cx, cy) et de rayon r, tourné d'un angle *angle*.

Modifiez le script `figures.php` pour ajouter le dessin du triangle.
Question 1.4 : Modifiez le script `figures.php` pour ajouter un argument obligatoire nommé `fig` dont la valeur attendue est `cercle, carre` ou `triangle`. Seule la figure choisie sera dessinée par le script. Un argument incorrect produira la page d'erreur.
Question 1.5 : Créez une page `formulaireDessiner.html` qui contient un formulaire permettant d'appeler le script `dessiner.php`.
Question 1.6 : Vous allez adapter le formulaire de façon à pouvoir dessiner plusieurs types de figures en même temps, Vous allez modifier le champ `select` :

- ajouter l'attribut `multiple="multiple"`
- modifier le nom : `name="fig[]"`

Maintenant le formulaire renvoie non pas une valeur simple mais un tableau avec toutes les valeurs selectionnées.
Il reste à adapter `dessiner.php` de façon à prendre en compte le fait que `$_GET['fig']` est maintenant un tableau.

Exercice 2 : Transférez le dossier `starwars` dans votre espace de travail et sur webtp. Ouvrez le fichier `starWars.html` avec un éditeur de texte. Visualisez-le également avec un navigateur .
Il s'agit d'un formulaire HTML qui permet de commander des articles à une association, le *club des fans de Star Wars*. Les articles sont des figurines de collection des personnages de Star Wars.

Question 2.1 : Mettez en commentaire les éléments `input` permettant de choisir une figurine. Remplacez-les par un élément `select` à sélection multiple (même nom de variable, mêmes textes affichés et mêmes valeurs renvoyées). Vérifiez le résultat.

Question 2.2 : Quand il est déclenché le formulaire charge une page `factureStarwars.php`. Le but de cet exercice sera de l'écrire.
Voici comment se calcule le montant de la facture ;

- Chaque figurine coûte 15 euros HT,
- Une réduction de 15% est consentie pour toute commande de 5 figurines ou plus.
- Les adhérents au club bénéficient d'une remise de 10% qui s'applique aux prix ci-dessus (donc cumulable).
- L'adhésion coûte 5 euros HT.
- Pour terminer, la TVA est de 20%, et les frais de port de 7,5 euros HT.

La facture fera apparaître la liste des figurines commandées, les prix HT et TTC ainsi que les coordonnées de l'acheteur (nom, adresse, etc...)
Une grande partie du code de génération de la facture vous est fourni

- une classe Facture, et des classe annexes.
- un script de lecture d'arguments `lectureArguments0.php`, version très provisoire.
- un script principal, à compléter.

Consulter les commentaires de la classe Facture pour comprendre comment on l'utilise et complétez le script principal `factureStarWars.php` de façon à obtenir un facture.
NB : dans cette question il n'est, **pour l'instant**, pas demandé de vérifier la validité des paramètres.

La vérification de validité des paramètres reçus est indispensable afin d'éviter un comportement anormal du script qui pourrait produire des résultats erronés ou incohérents. Et surtout, c'est un élément crucial de la sécurité des sites web.

La vérification faite au niveau du formulaire (premières questions de cette fiche) est un confort apporté à l'utilisateur et un plus dans l'ergonomie du site, mais **n'apporte aucune sécurité** : il est en effet toujours possible d'envoyer une requête au serveur sans passer par le formulaire. S'agissant de notre exemple, si on envoie une requête avec une URL comme celle-ci :

`factureStarWars.php?fig[]=KingKong&fig[]=SpiderMan& ....`

La facture affichera des figurines totalement inconnues du club des fans de Star Wars. Certes, pas de gros problème de sécurité sur cet exemple (peut-être une problème de ridicule !), mais un fonctionnement erroné du site.
C'est pourquoi, il faut toujours tester chacun des paramètres pour vérifier s'il correspond à ce que l'on attend.

Cas des variables « énumérées »

Quand une variable doit appartenir à un ensemble fini de valeurs, une solution consiste à construire un tableau ayant ces valeurs pour clés (on utilise le tableau comme une table de hachage). Supposons, par exemple, que l'on attende un argument *fruit* dont la valeur serait nécessairement *pomme, poire* ou *orange*. on peut utiliser une portion de code comme celle-ci :

```
$fruitsAutorises = array('pomme'=>TRUE,'poire'=>TRUE,'orange'=>TRUE);
if (! isset($_REQUEST['fruit']))
  throw new Exception("argument fruit non fourni");
else if (! isset($fruitsAutorises[$_REQUEST['fruit']]))
  throw new Exception("argument fruit {$_REQUEST['fruit']} invalide");
```

Quelques remarques :

- On aurait obtenu la même table en l'initialisant par

  ```
  $fruitsAutorises = array_fill_keys(array('pomme','poire','orange'),TRUE);
  ```

  l'utilisation de cette fonction permet un code plus compact quand les ensembles de valeurs sont grands.
- Notez que l'on n'utilise pas ici la valeur booléenne du tableau `$fruitsAutorises`, mais simplement le fait que la clé est définie. On pourrait utiliser ce tableau pour associer aux valeurs attendues non pas TRUE, mais une information utile à la suite du script. Par exemple, si l'on avait besoin d'une traduction en anglais :

  ```
  $fruitsAutorises = array('pomme'=>'Apple','poire'=>'Pear','orange'=>'Orange');
  ```

Utilisation des filtres

PHP propose une fonction d'import filtré des variables externes. Voici comment récupérer un paramètre entier nommé `nombreEntier` dans le tableau `$_GET` :

```
$n = filter_input(INPUT_GET, 'nombreEntier', FILTER_VALIDATE_INT);
 if ($n === NULL)
  throw new Exception("argument nombreEntier non fourni");
else if ($n === FALSE)
  throw new Exception("argument nombreEntier {$_REQUEST['nombreEntier']} invalide");
```

Nous venons d'utiliser le filtre prédéfini pour les entiers, nommé `FILTER_VALIDATE_INT`. Il existe d'autres filtres que vous pouvez découvrir dans la documentation PHP
Si 'nombreEntier' n'est pas fourni, la fonction `input_filter` renvoie NULL. S'il est fourni mais ne convient pas au filtre, le résultat vaut FALSE.
Certains filtres demandent un paramètre supplémentaire. C'est le cas de `FILTER_VALIDATE_REGEXP`. Voici comment on pourrait filtrer un argument attendu sous la forme d'une lettre suivie d'une suite de chiffres décimaux :

```
$n = filter_input(INPUT_GET, 'nombreEntier', FILTER_VALIDATE_REGEXP,
                  array('options'=> array('regexp'=>'/^[a-zA-Z][0-9]+$/'))
                 );
```

Note ; il s'agit ici de la notation des expressions régulières Perl qui présente quelques particularités. Pour une recherche par motif exact, commencer par `/^` et terminer par `$/`

Question 2.3 :

Écrire un nouveau script `lectureArguments.php` destiné à remplacer le précédent.
Vous vérifierez en particulier l'adhésion, la civilité, les noms des figurines, la validité du code postal. Vous vérifierez que les autres champs obligatoires sont non vides.
En cas d'erreur vous afficherez une page HTML d'erreur.

Question 2.4 :
Transformez le fichier contenant le formulaire `starWars.html` en `starWars.php`. Vous insèrerez une portion de code PHP permettant d'intégrer un message d'erreur. Le comportement sera le suivant :

- Si la variable `$error` est définie, alors elle est supposée contenir un message d'erreur qu'il faut afficher avant le formulaire, dans un paragraphe d'identifiant `errorMessage`.
- Sinon, ce paragraphe n'est pas créé.

Puis vous ferez en sorte que, lorsqu'une erreur est détectée, le script `factureStarWars.php` renvoie non plus la page d'erreur, mais le formulaire précédé du message d'erreur (indication : utiliser `require`).

Question 2.5 : Amélioration du formulaire
En l'état, il possible d'envoyer un formulaire incomplet. Complétez le code HTML pour faire en sorte que l'envoi devienne impossible en l'absence d'information dans les champs `nom, prenom, voie, cp, commune`

Question 2.6 : De même, contraignez le champ code postal pour n'y valider qu'une chaîne de 5 caractères Vous ferez en sorte qu'une chaîne invalide s'affiche en rouge.