Université Paris Diderot **Internet et outils**
L1 **Année 2018-2019**

TP n° 5

Formulaires

Remarque Comme pour les TP précédents, vous êtes encouragé(e)s à chercher les informations dont vous avez besoin auprès de sources fiables. Nous vous conseillons les trois adresses suivantes :
— `http://www.w3schools.com/html5/`,
— `http://www.w3schools.com/tags/tag_input.asp` et
— `http://php.net/manuel/fr`.

1 Objectifs du TP

Dans ce TP, nous allons apprendre à manipuler les formulaires, du simple affichage au contrôle des données saisies dans un champ.

Vous avez vu en cours deux méthodes pour transmettre les données saisies dans le formulaire : GET et POST. La méthode GET transmet ces données dans l'URI. Cette méthode est limitée : la taille de l'URI est limitée à un certain nombre de caractères par certains navigateurs. La méthode POST ne transmet pas les données au sein de l'URI mais dans la requête HTTP.

Attention : il peut sembler que la mèthode POST est plus adaptée à l'envoi de données sensibles (par example, mot de passe), puisque les données n'apparaissent pas directement dans l'URI. Néanmoins, la requête HTTP circule également en clair sur le Web. Pour les données sensibles, il convient d'utiliser le protocole HTTP sécurisé (`https`).

La méthode GET garde l'avantage pour le développeur car on peut l'utiliser sans formulaires puisque les données peuvent aussi être passées par des liens. En outre, les formulaires utilisant GET permettent à l'utilisateur de lier vers des requêtes du moteur de recherche, d'enregistrer leurs horaires de bus actualisés dans leurs favoris, etc.

N'hésitez pas à consulter les transparents du cours pour retrouver la syntaxe des formulaires (balises, attributs, etc.).

2 Mon premier formulaire

Exercice 1 — *Mon premier formulaire*

1. Créez une page `formulaire.html` contenant un formulaire dans lequel on demande à l'utilisateur d'envoyer à une page `affichage.php` avec la méthode `get` les informations suivantes (à vous de choisir le type d'input à utiliser pour chaque élément) :
   — prénom,
   — nom,
   — sexe,
   — mot de passe.
   Ajoutez aussi un bouton de validation et un bouton pour réinitialiser le formulaire. La page `formulaire.html` ne contient a priori pas de code PHP (d'où l'extension `.html`) et doit, bien entendu, être vérifiée par le validateur `w3c`.
2. Créez maintenant une page `affichage.php` qui affiche dans une phrase les valeurs associées aux clés `prenom` `nom` et `sexe` (pour cette dernière on peut utiliser, par example, une instruction conditionelle). Par exemple, si les trois valeurs sont, respectivement, `Jeanne`, `Dupont` et `femme`, le message affiché pourrait être :

   > Bienvenue Jeanne Dupont. Ton mot de passe est... oh attention, mieux vaut ne pas le montrer à tout le monde.

Regardez maintenant le résultat de votre voisin(e). Pouvez vous deviner le mot de passe entré, bien que celui-ci n'apparaissent pas dans le texte affiché ?

Exercice 2 — *Un peu moins de transparence*

Pour des formulaires plus importants, la méthode GET a un certain nombre de limites. Le passage de paramètres par la barre d'adresse restreint le type des paramètres ainsi que leur taille (imaginez une image). Le procédé est en outre inadapté à la transmission d'informations sensibles. La seconde méthode de passage de paramètres, POST, va permettre de mieux traiter ce genre de cas.

1. Transformez `formulaire.html` et `affichage.php` pour que la transmission se fasse par la méthode POST.
2. Observez les différences entre les deux méthodes lors de la validation du formulaire.

Remarque : Il existe un tableau associatif qui par défaut contient la même chose que l'ensemble des variables `$_GET`, `$_POST` (et `$_COOKIE` que nous n'utilisons pas encore). Il s'agit de `$_REQUEST`. En général, les méthodes GET et POST peuvent être combinés pour transmettre les différents paramètres d'une page en profitant des avantages des deux méthodes.

Exercice 3 — *Un* GET *fait maison*

1. Créez un fichier `traitement.php`. Écrivez une fonction PHP `passeTabPage` qui prend en argument une chaîne de caractères correspondant à une URI, et un tableau, et qui renvoie une adresse passant en argument GET tout le contenu du tableau à l'uri donnée. Par l'exemple, l'exécution du code suivant :

```
<?php
   $t=array("prenom"=>"Bob",
   "nom"=>"Leponge",
   "sexe"=>"homme",
   "motdepasse"=>"patrick");

   echo passeTabPage("affichage.php",$t);
?>
```

   Devra afficher :

```
affichage.php?prenom=Bob&nom=Leponge&sexe=homme&motdepasse=patrick
```

2. Écrivez une fonction PHP `lien` qui prend en argument deux chaînes de caractères `$text` et `$uri` et qui affiche le bloc HTML correspondant à un lien `$text` vers l'URI `$uri`.
3. Composez les deux fonctions précédentes pour afficher un lien vers la page `affichage.php` de l'exercice 1 avec, en argument GET, tout le contenu d'un tableau.
4. Modifiez la page `formulaire.html` de l'Exercice 1 et le fichier `traitement.php`, afin que la validation du formulaire redirige non plus directement vers la page `affichage.php` mais vers la page intermédiaire `traitement.php` contenant :
   — une liste avec un récapitulatif des données, sauf du mot de passe,
   — un lien "revenir en arrière" pointant vers la page `formulaire.html`,
   — un lien "continuer" pointant vers la page `affichage.php`.
   Bien entendu, les arguments GET devront être transférés à l'aide des fonctions précédentes.

3 Inscriptions aux modules

Dans cette section, vous allez créer un site pour que des étudiants puissent s'inscrire à des UE, en complétant les fichiers `formulaire.html` et `affichage.php` de la section précédente.

Exercice 4 — *Le formulaire*

1. Créez un fichier `cours.php`, et dedans définissez en PHP un tableau associatif (`identifiant => cours`), où `identifiant` est un identifiant de votre choix qui sera utilisé dans la phase de traitement des données, et `cours` est le nom d'un cours tel que vous voulez qu'il s'affiche sur la page d'inscription aux UE. Ce tableau doit contenir entre 5 et 10 éléments.
2. Dans ce même fichier, écrivez une fonction `generer_checkbox` qui prend en argument un tableau tel que celui que avez écrit dans la question précédente et qui renvoie du code HTML (dans une chaîne de caractères) correspondant à un morceau de formulaire HTML (sans les balises ouvrants et fermants `<form>` et `</form>`) avec une checkbox pour chaque cours du tableau.
3. Complétez votre fichier `formulaire.html` pour que l'etudiant puisse aussi ajouter son numéro d'étudiant (quel type d'input ?) ainsi que cocher les cours de son choix. Utilisez la fonction définie dans l'exercice précédent, à l'aide de la commande `include`. Pour cela faire il faut modifier l'extension de votre fichier. Les données seront envoyé directement au fichier `affichage.php` (sans passer par `traitement.php`).
4. Modifiez votre fichier `affichage.php` pour afficher en plus du nom de l'élève, etc. la liste des cours qu'il a choisi. Attention, vous devez avoir accès au tableau de cours défini dans l'exercice précédent.

4 Des formulaires qui ont du style

Exercice 5 — *Choisir le style d'affichage*

1. Modifiez le fichier `formulaire.php` pour organiser le formulaire dans un tableau de deux colonnes. Dans la première colonne il y aura les noms des champs, et dans la deuxième les champs du formulaire.
2. Créez trois styles d'affichage pour le fichier `affichage.php` (vous pouvez par exemple créer trois fichiers CSS différents) Par exemple un avec un fond vert et une police orange (pour les personnes avec un mauvais goût), un avec une police plus grande (pour les malvoyants) et un classique (sans style).
3. Ajoutez ensuite un menu déroulant dans `formulaire.php` qui permettra à l'utilisateur de choisir avec quel style il veut voir s'afficher les résultats dans `affichage.php`. Modifiez `affichage.php` pour qu'il respecte l'affichage demandé.
4. De manière plus générale, créez un fichier `style.php` qui contient un tableau avec tous les styles disponibles pour votre page. Ecrivez-y une fonction qui prend en argument un tableau de style et génère une liste déroulante des styles disponibles. Modifiez les fichiers `formulaire.php` et (si nécessaire) `affichage.php` pour que l'utilisateur puisse choisir parmi tous les styles disponibles.

5 Un formulaire complet

Les éléments des formulaires HTML peuvent être très variés. Vos ressources préférées vous fourniront toutes les valeurs que peut recevoir l'attribut `type` de la balise `<input>`. D'autres attributs comme `value` et d'autres balises comme `<select>`, `<textarea>` et `<label>` vous seront également utiles.

Exercice 6 — *Des entrées de tous types*

Complétez vos pages `formulaire.php` et `affichage.php` pour en faire un formulaire de renseignement complet. Il devra demander les informations suivantes :
— Prénom (type `text`),
— Nom (type `text`),
— Date de naissance (type `date`),
— Numéro d'étudiant (type `number`),

— Sexe (choix exclusif),
— Courriel (type `email`),
— Téléphone (type `tel`),
— Mot de passe (type `password`),
— Confirmation du mot de passe (type `password`),
— Outil préféré : HTML, PHP, ou CSS (liste déroulante),
— Vous aimez ce cours ? `oui` ou `non` (choix exclusifs, valeur par défaut `oui`),
— Liste des cours auxquels vous êtes inscrit (options non exclusives à cocher comme dans l'Exercice 5),
— Qu'aimeriez-vous faire ensuite ? `un forum`, `un site de partage d'images`, `un réseau social` (options non exclusives à cocher, noms des champs `projet1`, `projet2` et `projet3`, une case de votre choix cochée par défaut),
— Commentaires (zone de texte active de 40 colonnes et 10 lignes),
— Style d'affichage (liste déroulante comme dans l'Exercice 5),
— Un bouton de réinitialisation avec l'étiquette "Annuler" (bouton de type `reset`),
— Un bouton de validation avec l'étiquette "Valider" (bouton de type `submit`).

Exercice 7 — *Vérification des saisies*

Il se peut que les utilisateurs des formulaires ne saisissent pas les données exactement comme vous les voulez. Les fonctions `htmlentities` et `htmlspecialchars` vous prémunissent contre les utilisateurs malveillants, mais même les utilisateurs bienveillants peuvent faire des erreurs : oublier l'arobase dans leur adresse email, taper un mot de passe différent la seconde fois, rentrer du texte là où vous attendiez un nombre, etc. Il faut donc constamment s'assurer que les valeurs présentes dans $_GET ou $_POST soient de la bonne forme. Sinon, vous subirez des erreurs PHP, ou pire, des conversions silencieuses qui généreront des bugs bien plus tard.

1. Modifiez le fichier `affichage.php` afin qu'il ne se contente plus de retranscrire brutalement tous les paramètres qui lui sont passés. Il faut notamment vérifier les choses suivantes :
   — les champs essentiels (à vous de décider lesquels) ont bien été remplis (`isset`) (pensez à modifier aussi le fichier `formulaire.php` pour prévenir l'utilisateur),
   — les deux occurrences du mot de passe sont identiques,
   — on ne peut pas écrire de code HTML dans le formulaire (`htmlspecialchars`).
   Faites en sorte qu'en cas de problème, la page `affichage.php` affiche un avertissement et un lien vers la page `formulaire.php`.

6 Partie otpionnelle

6.1 Un véritable traitement des données

Jusque là, nous avons seulement affiché telles quelles les données du formulaire. Dans l'exercice qui suit, nous allons faire quelques calculs avec les données de l'utilisateur.

Plus tard, nous aurons des outils pour stocker l'information (les bases de données). Pensez à la situation la plus courante dans laquelle vous rencontrez un formulaire HTML : votre inscription sur un site web. Vous tapez le pseudo que vous désirez utiliser, ainsi qu'un mot de passe. Le serveur récupère ces informations, et doit les stocker afin de pouvoir vérifier la combinaison pseudo/mot de passe lors de vos connexions ultérieures.

Exercice 8 — *Résultats du bac*

1. Créez une page `notes.html` avec un formulaire qui demande les notes au bac des matières suivantes : Français, Maths, Histoire-géo, Anglais et EPS.
2. Créez un fichier `bac.php` qui doit afficher les notes précédentes en deux listes séparées (une pour les notes supérieures ou égales à 10, une pour les notes strictement inférieures). Vous écrirez cette portion de code dans une fonction `affiche_notes($notes)`

3. En réalité les seuils de séparation des notes peuvent varier selon ce qu'on souhaite regarder En effet dans certains cas, une note entre 7 et 10 pourrait donner lieu à un rattrapage, une note inférieure à 7 serait éliminatoire, et une note inférieure à 2 pourrait donner lieu à un entretien. Modifiez votre fonction d'affichage pour mettre en valeur ces différents seuils. Pour cela vous pourrez écrire une fonction `filtre_valeurs` qui prendrait en argument un tableau de valeurs `$v`, une valeur de seuil `$lim`, et un sens `$sens` de filtre. Il retournerait alors un tableau composé des valeurs de `$v`, supérieures ou égales au seuil `$lim` si `$sens` est positif (ou nul) et strictement inférieures à `$lim` si `$sens` est négatif.
4. Améliorez votre script, en écrivant une fonction dans `bac.php` qui se charge d'afficher un message d'erreur et un lien ramenant à la page `notes.html` dans le cas où l'une des notes rentrées par l'utilisateur n'est pas un nombre (fonction `is_numeric`) compris entre 0 et 20.
5. Pour finir, écrivez des fonctions `moyenne($notes)` et `mention($n)` pour calculer la moyenne générale (vous pouvez attribuer des cofefficients arbitrairement) et convertir ce résultat en mention (recalé, rattrapage, passable, assez bien, bien ou très bien).

6.2 Pendu

On va à présent créer un jeu de Pendu en PHP.

Exercice 9 — *Utilisation de fonctions natives*

1. Dans une page `outils.php`, écrivez une fonction `contient($lettre,$alphabet)` qui prend en argument la chaîne de caractères `$lettre` représentant une lettre et une chaîne de caractères `$alphabet` représentant un ensemble de lettres, retourne `true` si la lettre apparaît dans l'alphabet et `false` sinon. Pour ce faire, vous pouvez utiliser la fonction native `strlen` qui calcule la longueur d'une chaîne de caractères.
2. Écrivez une page `pendu.php` prenant deux paramètres (pensez à utiliser le tableau `$_GET` ou le tableau `$_REQUEST`) :
   — un paramètre `mot` qui contiendra le mot à deviner,
   — un paramètre `lettres` qui contiendra les lettres que le joueur a déjà essayé.
   La page affichera le contenu de `mot` en remplaçant les lettres qui ne sont pas présentes dans `lettres` par des tirets bas.
   Par exemple, la page `pendu.php?mot=journal&lettres=aoub` doit afficher le mot `_ou__a_`.
3. Écrivez une fonction php `compte_lettres` prenant en paramètres deux chaînes de caractères, et qui renvoie le nombre de lettres de la seconde chaîne qui n'apparaîssent pas dans la première.
4. En utilisant la fonction `compte_lettres`, faites en sorte que la page affiche le texte « Vous êtes pendus ! » lorsque le joueur a essayé plus de 10 lettres qui n'apparaissent pas dans le contenu de `mot`, et « Vous avez gagné ! » si le mot est complètement dévoilé.
5. Rajoutez à la fin de la page `pendu.php` un formulaire composé d'un champ de saisi de texte associé à un bouton de validation qui va reçevoir la nouvelle lettre à jouer.
   Il existe en html des formulaires de type `hidden`. Ces formulaires permettent de définir des champs qui ne seront pas affichés chez l'utilisateur. Les champs cachés peuvent contenir des données et lors de l'envoi d'un formulaire, les données de champs cachés sont elles aussi transmises. Par ce moyen, des informations supplémentaires peuvent par exemple être transmises à des programmes.
   En utilisant des formulaires cachés, faites en sorte que, lorsque le joueur saisit une nouvelle lettre, il soit envoyé sur une nouvelle page `pendu.php` où l'argument `mot` est le même que celui de la page de départ, et où l'argument `lettres` est constitué de son ancienne valeur, concaténée avec la lettre rentrée par l'utilisateur.
6. Jusqu'à présent, le joueur peut tricher en regardant dans l'URI la valeur du mot à deviner. Pour éviter cela, on va utiliser un fichier de texte.
   Créez un fichier texte `mot_secret.txt` contentant une chaîne de caractères. A l'aide de la fonction `verb+file_get_contents("nomdefichier.txt")` de PHP, faites en sorte que le mot à deviner soit le contenu du fichier `mot_secret.txt`.

7. Créez maintenant un fichier texte `liste_mots.txt` contenant une liste de 100 chaînes de caractères. Modifiez le fichier `pendu.php` pour que si `mot` n'est pas initialisé, alors un mot est choisi au hazard dans la liste `liste_mots.txt` (pensez à utiliser les fonctions natives `rand` et `explode`).